SIGNAL SOURCE SEARCHING AND TRACKING VIA ANTENNA ARRAY BEAM STEERING ON FIXED WING UAVS FOR COMMUNICATION LINK ENHANCEMENT

BY

Mohamed Adel Ibrahim

A Thesis Presented to the
DEANSHIP OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

In

ELECTRICAL ENGINEERING

December 2012

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN- 31261, SAUDI ARABIA

DEANSHIP OF GRADUATE STUDIES

This thesis, written by **Mohamed Adel Ibrahim** under the direction of his thesis advisor and approved by his thesis committee, has been presented and accepted by the Dean of Graduate Studies, in partial fulfillment of the requirements for the degree of **MASTER OF SCIENCE IN ELECTRICAL ENGINEERING.**

Dr. Mohammad S. Sharawi
(Advisor)

Dr. Ali A. Al-Shaikhi
Department Chairman

Dr. Ahmad A. Masoud
(Member)

Dr. Salam A. Zummo
Dean of Graduate Studies

Dr. Azzedine Zerguine,
(Member)

29/1/13

Date

Dedication

"And do not forget liberality between yourselves. For Allah sees well all that ye do."
Al-Baqarah 237

This work is dedicated to my affectionate mother, for whom I should say, thank you for raising me in the best way. Thanks for helping and encouraging me to be the man I am today. Thank you for teaching me love, understanding, and respect. Thanks for always being there for me.

Thanks to my judicious father, he will always be my favorite mentor. Thanks for teaching me how to be free and independent. Thanks for guiding me to the truth. Thanks for encouraging me to read a lot.

Thanks to my gallant older brother for always being my friend and for keeping my secrets. He will always be my lead in life.

Thanks for my beloved wife. I wouldn't have made it without her love, care, and understanding. Thank you for encouraging me, and for keeping me happy. Thank you for being on my side all the way from the beginning, and for taking me for who I am. Also I would like to thank her family for raising such a wonderful woman and for trusting me to take care of her.

I present a special dedication to my beautiful daughter. I wish her happiness in life and a special place in heaven.

ACKNOWLEDGMENTS

I would like to show full gratitude to my energetic active advisor, Dr. Mohammad Sharawi, for his constant moral, financial and scientific support as well as for always being there for me. Also thanks to my team mates in the research group of Dr. Sharawi for providing thoughts and comments throughput the group meetings.

Also I want to thank my committee members Dr. Ahmad Masoud and Dr. Azzedine Zerguine, for their valuable feedback throughout the research process. Additional Thanks to Dr. Mohammed Khalil Ibrahim and Dr. Mohammad Abido for Their valuable comments.

I present special thanks to Dr. Mohamed Ali Hassan for guiding me on the professional level and the personal level as well. Another special thanks to my colleague Mr. Sameir Deif for being a good friend and a powerful team member.

I acknowledge the department of electrical engineering at King Fahd University for petroleum and minerals for providing a very good work environment.

Finally I would like to thank all my colleagues and friends and I wish them a happy and fruitful life.

Table of Contents

List of Tables

List of Figures

English Abstract

Full Name: Mohamed Adel Mahmoud Mohamed Ibrahim

Title of the study: Signal Source Searching and Tracking via Antenna Array Beam Steering on Fixed Wing UAVs for Communication Link Enhancement.

Major Field: Antennas and signal processing.

Date of Degree: December 2012.

Unmanned air vehicles (UAVs) have been given a lot of attention recently, because they proved to be very useful in a lot of applications in civil and in military sectors. These applications include, but not limited to remote sensing, surveillance, exploration, and search and rescue.

In this work we present the design of a patch planar antenna array, which will be installed inside the wing structure of a hobby type mini-UAV. The array has the ability to steer its beam with two degrees of freedom in spherical space (θ, ϕ). Two array configurations are considered for simulations to characterize their radiation pattern, half power beam width (HPBW), and their pointing gain. The maximum gain for the 6×2 array is $22\ dB$ and for the 7×2 array is $24\ dB$. The array was fabricated and

tested. The S-parameters were measured to validate the isolation between adjacent elements.

A beam steering algorithm based on measuring the received signal strength (RSS) is proposed to control the antenna beam and steer it to the maximum RSS in real time as the UAV is flying. The proposed elliptical Peeking (EP) algorithm is simulated and compared to a well-known intelligent optimization algorithm called the differential evolution (DE). A performance assessment is presented for two types of flight trajectories, smooth and turbulent. The EP algorithm showed better performance where it was able to close on the maximum RSS in $450\ ms$ on average.

A virtual UAV trajectory is created to be used in testing the performance of each algorithm. Two methods are used to create the trajectory. The first is based on random variables, and the second is based on a flight simulator. The second method has proved its validity and feasibility while the first method did not.

A tracking routine is proposed to keep locking on the direction of the maximum received signal strength. Its performance is assessed in terms of signal degradation for two types of flight trajectories, smooth and turbulent.

The EP algorithm was used to conduct a new search round every fixed time period.

This work presents a system that integrates the search and tracking algorithm within an antenna beam steering based on received power levels to enhance the communication links of flying UAVs.

ملخص الرســـــالـــــة

الإسم: محمد عادل محمود محمد ابراهيم.

عنوان الرسالة: البحث عن مصدر الإشارة و تعقبها باستخدام شعاع موجه لمصفوفة هوائيات مثبتة على طائرة بدون طيار مثبتة الجناح من أجل تحسين الإتصال اللاسلكى.

التخصص: الهوائيات و معالجة الإشارات.

تاريخ التخرج: ديسمبر 2012

لقد جذبت الطائرات بدون طيار كثيراً من الإهتمام فى الآونة الأخيرة لأنها أثبتت فعاليتها فى كثير من التطبيقات فى القطاعات المدنية و العسكرية. هذه التطبيقات تشمل و لا تنحصر فى الإستشعار عن بعد، المراقبة، الإستكشاف، و البحث و الإنقاذ.

فى هذا العمل البحثى نقدم تصميم مصفوفة هوائيات مسطحة مطبوعة. حيث ستركب داخل جناح طائرة بدون طيار من الطراز الصغير المستخدم من قبل الهواة. المصفوفة لديها القدرة على توجيه شعاعها فى الفراغ بدرجتين من الحرية. تم إعتبار شكلين من المصفوفة و محاكاتهما لدراسة التوزيع الإشعاعى، عرض الشعاع عند نصف القدرة، و الكسب النسبى فى نقطة التوجيه. أقصى كسب نسبى للمصفوفة 2×6 كان $22\ dB$ و للمصفوفة 2×7 كان $24\ dB$. تم صنع المصفوفة و إختبارها فى المعمل وقياس عوامل الإنتشار للتأكد من العزل الكافى بين العناصر المتجاورة فى المصفوفة.

تم إقتراح خوارزم لتوجيه الشعاع معتمداً على قياسات قوة الإشارة للتحكم فى شعاع المصفوفة و توجيهه لأقوى إشارة فى الوقت الفعلى أثناء طيران الطائرة. تم محاكاة الخوارزم المقترح (إختلاس النظر بالقطع الناقص) و مقارنة النتائج بخوارزم ذكى معروف مسبقاً (التطور التفاضلى). تم تقديم دراسة لكفاءة الخوازمين مستندة على نوعين من المسارات للطائرة و هما المسار الإنسيابى و المسار

المضطرب. خوارزم إختلاس النظر بالقطع الناقص قد تفوق على خوارزم التطور التفاضلى حيث أنه إستطاع العثور على مصدر الإشارة فى $450\ ms$ فقط.

تم إستخدام طريقتان لإنشاء مسار إفتراضى للطائرة لإستخدامه فى تقييم كفاءة الخوارميات. الطريقة الأولى إعتمدت على المتغيرات العشوائية و الطريقة الثانية إعتمدت على محاكى طيران. الطريقة الثانية أثبتت صلاحيتها على عكس الطريقة الأولى.

تم إقتراح أسلوب تعقب من أجل الحفاظ على إتجاه الشعاع دائماً نحو أقوى إشارة بدون أن تتأثر عملية التوجيه بحركة الطائرة و مناوراتها. تم تقييم كفاءة أسلوب التعقب من حيث تراجع قوة الإشارة فى نوعى المسارات المستخدمة للطائرة (الإنسيابى و المضطرب). تم إستخدام خوارزم إختلاس النظر بالقطع الناقص لتنفيذ عملية بحث كل مدة محددة.

هذا العمل يقدم نظام متكامل ما بين البحث و التعقب للإشارات اللاسلكية باستخدام مصفوفة هوائيات معتمداً فقط على قياسات قوة الإشارة الواردة لتتحسين جودة الإتصال بين الطائرة و محطتها الأرضية.

Chapter 1: Introduction

Unmanned aerial vehicles (UAVs) have attracted a lot of attention recently due to their use in several applications in civil and military sectors. While military applications hold the biggest share of interests in UAVs; they are finding their own way into civil and scientific applications like exploration, mapping, search and rescue, remote sensing, and even entertainment.

Since these vehicles are unmanned, they must have a degree of autonomy in performing their appointed tasks. However, even in fully automated vehicles it is preferred to have a direct access communication link to the vehicle in order to supervise its behavior, change its target, receive its sensory feedback, abort its mission, or perform any unplanned action in real time which in some cases is considered crucial.

The real time communication between the vehicle and its controlling station is divided into two different links, the control link and the data link. The control link[1] usually has a long operating range due to its relatively low frequency of operation, while the data link[2] has a short range due to its higher frequency of operation (i.e. 2.45 GHz industrial scientific and medical -ISM- band) and it is used in transmitting sensory data to the ground station.

Increasing the range of the data link can be achieved using an antenna array within the UAV. The array will generate more signal gain and thus increases the link range. If the antenna is embedded within the UAV structure the aerodynamic profile of the vehicle

[1] Control link: Uplink channel.
[2] Data link: Downlink channel.

will be improved, where it will reduce the drag force and will increase the battery life. Since the antenna will be installed inside the vehicle's structure. This will improve the reliability and the life time of the antenna because it will be installed in a less hostile environment.

Having a directed focused beam from the UAV will improve the communication link only if the beam is directed towards location of the ground station, and that is why localizing and tracking the signal of the ground station are needed. This localization can be achieved by using a global positioning system (GPS) receiver since everything is carried outdoors. However as far as system security and independence are concerned, the GPS signal can be easily jammed which will dramatically degrade the link stability and reliability. Also it is not always guaranteed that the direction of the ground station is the direction of the maximum received signal strength (RSS), especially if we are dealing with reflected signals when a direct line of sight (LOS) is unavailable.

Radio signal source localization is always based on a developed algorithm which depends on one or more measurable signal parameters. The most common of these signal parameters are the angle of arrival (AOA), the time of arrival (TOA), the time difference of arrival (TDOA), and the RSS. Each one of these signal parameters has its own computation method and hardware requirements. This makes certain parameters suitable in some situations and unsuitable in others, thus we shall look into this issue based on the problem at hand.

The search and tracking algorithm which will control the antenna beam might require large amount of computational resources, and in turn this will require a complex

hardware platform. Thus we tried to maintain the algorithm simple in terms of mathematical complexity.

Some parameters in the proposed algorithm and the antenna array were set and simulated according to the specifications of some commercially available hardware and a specific commercially available UAV. Thus the algorithm must be tuned before using it on a different hardware platform and the antenna must be modified before mounting it on any other UAV.

This work is composed mainly of three phases:

1- Antenna array design and characterization.
2- Search and track algorithm development based on RSS measurements.
3- Integrating the system components and assessing the system overall performance in accomplishing the desired beam steering based on a flying UAV platform.

1.1 Previous work

In this section we will present a general overview of the previous works on the system level. Later within individual chapters we will present the previous works focusing on the topic addressed in the chapter.

In this section our review includes the systems performing signal source localization and radio direction finding. The architecture of these systems depends greatly on their carrying platform. Different platforms include stationary ground, mobile ground, and flying platforms. We will focus in our review on the systems proposed for flying platforms (UAVs) only.

Research in UAV related applications both for military and civil use has covered a broad number of fields such as control, wireless communications, signal processing, and antenna design.

Since more sophisticated UAVs are more autonomous, they have to be independent from direct control commands coming from an operator, hence path planning and standoff tracking becomes critical and it has been studied in a number of articles such as [1]–[3].

One of the most important uses of UAVs is visual inspection by which an operator can collect information about his surrounding area as well as use these video streams to feed an image processor in order to identify ground objects and track them [4]–[6]. In [7] the UAV trajectory was optimized for a ground target localization mission with the help of an image sensor. However, visual inspection limits the UAV range in which it can localize and track targets due to resolution constraints of image sensors as well as visibility constraints caused by weather.

Another form of localization and tracking of targets is based upon electromagnetic signals. The work in [8] describes a technique to vary the stationary temporal window of the target in order to improve the tracking accuracy of the tracking filter which is fed by a direction of arrival (DOA) sensor [9], [10]; however, other signal sensing methods can be used such as angle of arrival (AOA) [11], and received signal strength (RSS) [12].

RSS based systems are capable of localizing and tracking targets indoors [12]–[14] as well as outdoors [12]; however, indoor problems get more attention because usually outdoor problems are easily solved by using a GPS receiver [15] which is incapable of operating accurately indoors due to poor signal reception.

For a UAV it has been proven that embedding an antenna array inside the UAV's structure is more efficient in terms of payload maximization and air drag minimization [16], [17]. This also will increase the battery life time. Since we are using an antenna array embedded in a UAV structure, it is possible to control the phases feeding the array in order to steer the beam towards the ground transmitter which will certainly improve the communication link between the UAV and the ground station. This will be discussed in more details in Chapter 3.

Signal source tracking was tackled in [18] where they used a mono-pulse tracking technique to keep the beam pointing to the direction of the received signal, other tracking methods are used as well. Chapter 5 gives a deeper insight for these methods.

Although tracking is a well tackled problem in the literature, all tracking algorithms must be provided with an initial estimate. This incorporates the necessity for a signal search algorithm to provide this initial estimate for the tracking filter. It is worth mentioning that the more accurate the initial estimate the better the tracking accuracy regardless of the type of the tracking Algorithm. Also any tracking filter must be constantly fed by a location measurement in order to keep track of its target. Huber [9] has some interesting insights about signal direction finding, however when it came to the hardware implementation he used a bulky radio direction finding (RDF) device which cannot be integrated in a small UAV. Figure 1.1 depicts this device.

Figure 1.1 An example of a dual-dipole type homing RDF unit [9].

In [19] Bayes inference was used to achieve better localization for a 121.5 MHz emergency locating transmitter (ELT), using a number of fixed receiving stations shown in Figure 1.2 and an RDF device operating in each station. This method of course relies on multiple fixed transmitter locations and will be practical for ground stations and not for a system with a flying UAV.

Signal source searching can be based upon several types of signal measurements, we will incorporate only a received signal strength indicator (RSSI) for our signal searching and tracking in this work. It is worth mentioning that there is an emphasis in literature towards localization problems, and most of it is dedicated to localization in wireless sensor networks (WSN) [20] and wireless networks [21].

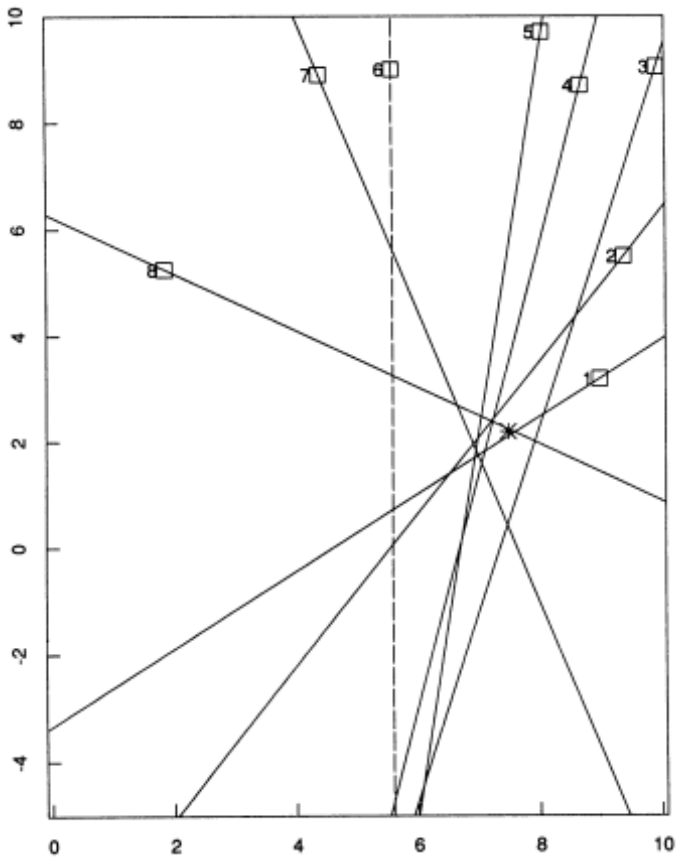

Figure 1.2 Locations of stations in length (1981), the lines represent observations. The bearing at station 6 is used for sensitivity analysis only. The asterisk represents the target ELT [19].

1.2 Thesis organization

The thesis is organized and presented in 7 chapters. In Chapter 1, we introduce the topic and conduct a general literature review by surveying the problem at the system level, and we state the thesis contributions. In Chapter 2, we formulate the problem in a structured organized manner and we describe our assumptions. In Chapter 3, the antenna array design and characterization by both simulations and measurements are presented. In Chapter 4, we present the procedures we went through in generating the virtual trajectory needed for testing our search algorithm. In Chapter 5, the search algorithm along with its simulation results are discussed. We also compare the obtained results from the proposed

algorithm with an evolutionary optimization method. In Chapter 6, we describe the tracking routine and present its performance results in simulation. In Chapter 7, the thesis is concluded and our vision for possible future work on the topic is presented.

1.3 Thesis contributions

The contributions of this work are:

1) Designing a light weight patched planar antenna array to be embedded within the wing structure of a UAV that operates at $2.45\ GHz$.
2) Creating a virtual environment for testing the beam steering algorithm.
3) Introducing an RSS beam steering algorithm (elliptical Peeking algorithm) to search for the signal source.
4) Providing a performance comparison between the proposed search algorithm and a well-known evolutionary optimization algorithm.
5) Developing a tracking routine based on the proposed elliptical Peeking algorithm. This routine doesn't involve any tracking filters or any computationally complex procedures.

A block diagram showing the system level components is shown in Figure 1.3. The shaded blocks represent the contributions of this work.

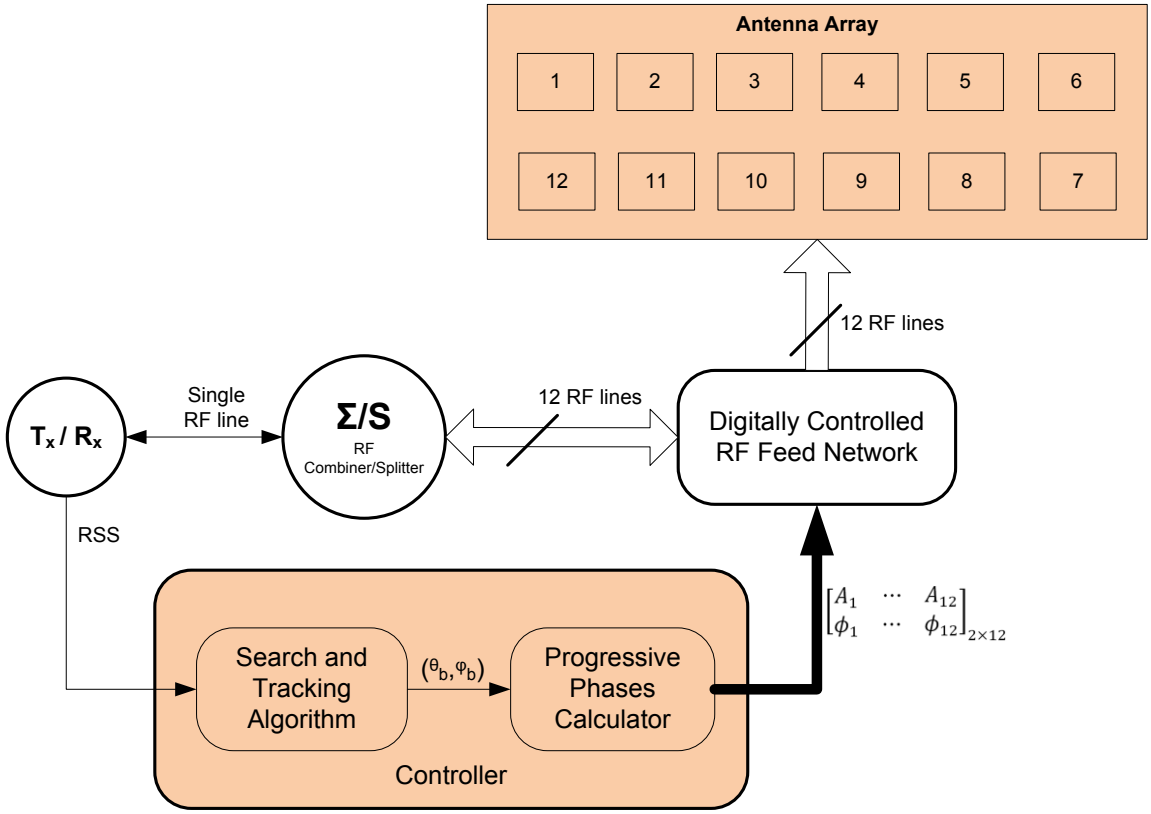

Figure 1.3 A Block diagram showing the whole system, the shaded blocks are the contributions of this work.

Chapter 2: Problem Formulation

In this chapter we will describe the problem at hand in full details and we will present the assumptions. Also an overview of our methodology will be provided. These issues should be enough to provide the reader with the full picture about the problem.

2.1 Problem Description

For a given fixed wing UAV connected wirelessly to its ground control station, we are attempting to provide this UAV with the ability to search for the strongest signal coming from its ground station and track this signal as the UAV is in motion in order to provide a better communication link for data transmission, hence increasing the range of operation between the UAV and the ground station. Figure 2.1 shows a successful search case where the vehicle's antenna beam is steered towards its ground station.

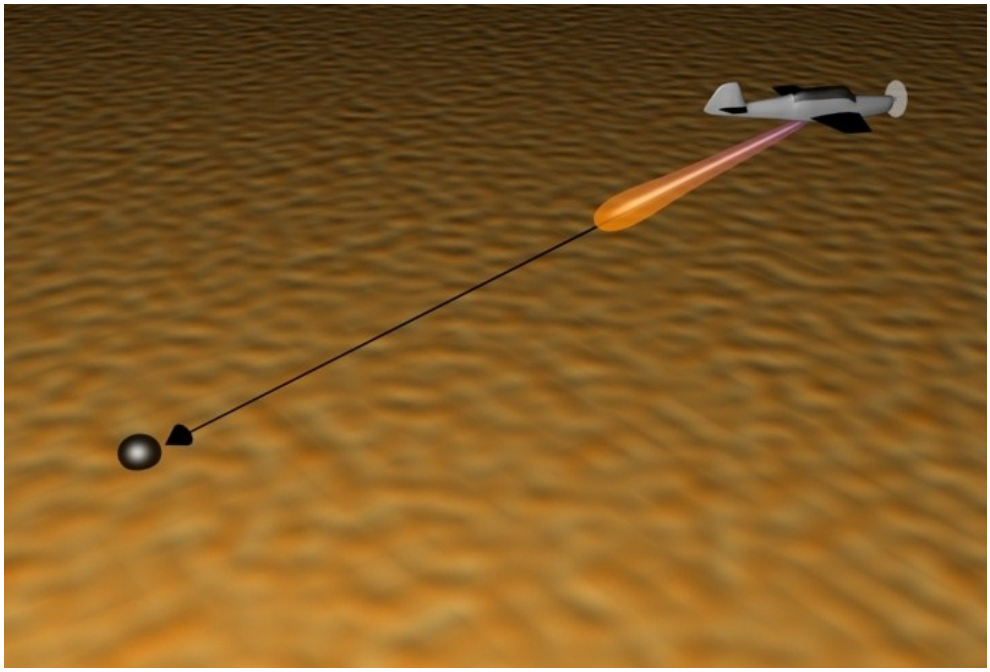

Figure 2.1 a 3D computer generated model showing a flying UAV with its antenna beam is pointing towards the location of its ground station.

To be able to steer the beam towards the transmitting ground station the first task would be to search for the strongest signal which should be performed using the minimum amount of hardware and consuming minimal time with an acceptable level of accuracy.

After the completing the search; estimated elevation and azimuth angles will point the beam towards the direction of the maximum RSS via a beam steering algorithm that will provide the appropriate phase excitations to the antenna array elements. After executing the search algorithm we would like to track the signal of the ground station, also using the minimum amount of hardware effort and with an acceptable level of accuracy to keep the beam directed towards the ground station. This will help in having a better communication link for data transmission. The vehicle will use a passive sensing procedure by measuring the RSS (received signal strength) for any given beam direction, and this shouldn't be confused with passive RADARs.

Searching and tracking are used to keep the beam of the antenna array pointing towards the ground station. Thus a third task would be to design an embedded antenna array that can be steered towards the ground station according to the directions identified by the search and the track modules. A planar antenna array will be used to allow for better beam steering towards a specific azimuth and elevation angles. It is considered good enough in our case if the direction of the highest RSS is caught within the -3 dB footprint of out planar antenna array. Thus the -3 dB would be our target for the maximum allowable signal degradation due to beam misalignment.

2.2 Methodology

In this work we will present an embedded antenna array within the UAV structure that is capable of tracking the location of the ground station with the aid of search and tracking algorithms for communication link enhancement.

In order to model and simulate the various portions of this system we have to generate a virtual trajectory for the UAV and use it in testing our algorithms. Figure 2.2 shows the flow chart of our main procedures in the simulation process through which we will evaluate the algorithm and assess its accuracy.

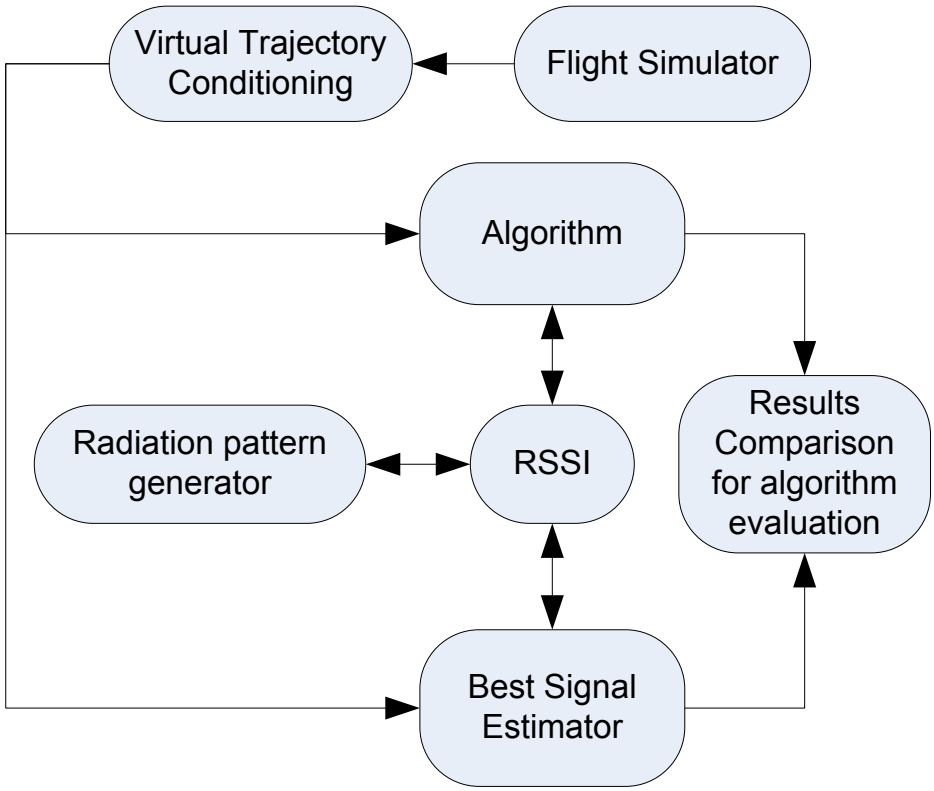

Figure 2.2 flow chart of the main procedures in the system simulation model.

In the flow chart shown in Figure 2.2, the simulation starts by piloting a virtual aircraft while generating a flight log containing all the aircraft position and attitude parameters, this is achieved by running an open source flight simulator. Then the virtual trajectory is interpolated and conditioned for matching the time step between successive RSS measurements which is 5 ms according to the typical time delay between successive RSS readings stated by the receiver's specifications [22]. The algorithm conducts a series of RSS measurements which is why we need an RSSI module. In order for the RSSI to calculate its output, a signal propagation model is used which requires the value for the antenna gain. The antenna gain is calculated by the radiation pattern generator according to the vehicle position, the vehicle attitude, the ground station position, and the current steering angles given by the algorithm. The best signal estimator assumes a perfectly steered beam and calculates the strongest signal to be achieved according to the current situation of the vehicle. Then the results of the algorithm and the best signal estimator are compared where now the recorded difference in the RSS is only due to the misalignment between the beam produced by the algorithm and the direct line of sight between the vehicle and its base station. This simulation is repeated for a large number of different trajectories in order to assess the algorithm performance extensively.

Each one of these blocks will be described in details in later chapters for a deeper understanding of how the simulation environment was created.

2.3 Problem Assumptions

In this work, several assumptions are used for both the environment and the UAV. These assumptions are:

1- Clear good weather (No rain, No sand storms, No elevated humidity levels). This is because any of these weather conditions requires a signal propagation model different from what we used in this work.
2- Direct line of sight (LOS) between the vehicle and the ground station, even in case of signal reflection, the path of the maximum RSS will be searched and tracked.
3- A stationary ground station with an isotropic radiator.
4- The ground station in continuously transmitting wireless signals, so that RSS readings at the UAV will be available at any time.
5- The proposed receiver is an XBee-pro ZB Module [22] whose specifications are:

   Receiver sensitivity: $-102\ dBm$

   Transmit power: $10\ dBm$

These assumptions are considered and used in the virtual trajectory generator and the search and tracking algorithm simulations as they represent the real constraints introduced by the hardware modules as well as the environment. The hardware modules of interest are the communication module (XBee), the UAV structure for antenna array design, and the computing hardware core.

Chapter 3: Embedded Antenna Array design

3.1 Introduction

Antenna design is one of the major factors that affect the performance of any wireless communication system. It is constrained by a number of factors that are not all related to its electromagnetic profile, such as mechanical size, weight, and cost.

In our design of an antenna array for UAV wireless communication link enhancement, it is important to consider its effect on the aerodynamic profile of the UAV since the designed array is going to be installed on the body of a flying vehicle. One of the main reasons of embedding an antenna array within the structure of the UAV is to reduce its effects on the aerodynamic profile of the UAV. Another reason is to allow for an increase in the vehicle's payload capability due to the reduced weight of the antenna structure when it becomes part of the structural components of the UAV. Antenna arrays maximize the gain in a certain direction for enhancing the received signal to noise ratio, hence enhancing the UAV's communication range.

Our Objective in this chapter is to design a planar antenna array which can fit easily inside the wing of a small size UAV (mini-Telemaster [23]) and light enough to cope with its low payload capabilities. We optimized our design to work in the ISM frequency band, specifically at 2.45 GHz.

3.2 Previous work

Antennas are an essential component in any unmanned air system (UAS). There are a lot of choices to make when we are designing an antenna for a UAV. Some of the antenna

properties such as weight and size must be tuned to match the capabilities of the vehicle such as its payload and its structure.

Embedding the antenna inside the wing structure is useful for both the antenna and the vehicle. Where for the antenna it will be installed in a less hostile environment which will improve its reliability and lifetime, and for the vehicle it will reduce its drag force due to air friction. This has been attempted by several previous works wherein [16] they designed a 4-element linear array operating at 2.4 GHz, and they implemented the array as a part of the mechanical structure of a small UAV.

In [17], three different antenna types were studied, two of them were not parasitic, a slot antenna covered with a radome (radar dome) and a dipole antenna. The third one was a parasitic antenna. Since these antennas were single elements, they had static radiation patterns. Thus beam steering was not available in this work.

In [24], a number of antenna elements were studied but their combinations were limited to single elements or linear arrays. Thus beam scanning was available with only one degree of freedom. This was also the case in [25]-[18].

In [26] a wide band planar array was developed that worked at X/Ku frequency band (9.2-10.4 GHz) which is way higher than our frequency of interest (ISM band, 2.45 GHz). Antennas designed for a certain frequency band are not expected to radiate intentionally in other frequency bands. Also the planar array produced showed the ability to produce multiple beams, yet it had poor angular resolution compared to system presented in this work.

In [27] a 4×2 microstrip patch planar array was developed and installed in a UAV to generate topographic maps for earth science. However the operating frequency was

1.2575 GHz with 80 MHz bandwidth and only azimuth scanning was available with a resolution of ±20˚ which is considered as a very low scanning resolution for our application.

The antenna array designed in this work is accompanied by a digitally controlled feed network that will allow for more scanning resolution. Also, the number of elements can be adjusted to fit any small size UAV. It is light weight and easily manufactured with a very low cost. Moreover, it can be utilized by UAV designers to be a part of the mechanical structure due to the rigidity and flexibility of the FR-4 material used. This can be easily achieved by extending the dimensions of the FR-4 material without extending the ground plane or adding any extra radiating elements. Thus without affecting the radiation pattern, mechanical designers can have enough space for any mechanical attachments (Bolts, screws … etc.). On the other hand it is recommended to use glue or non-metallic mechanical links to attach the antenna in order to prevent any deformation in the radiation pattern.

3.3 Design

In this section the design and implementation of a planar printed antenna array which will be embedded in the UAV wing structure is presented. Its gain and half power beam width versus the number of elements are investigated. The improvements in the system communication and range metrics are evaluated as well. A beam forming algorithm is used to control the beam heading direction with two degrees of freedom, where (θ_b and ϕ_b) are the two angles used to steer the beam in a spherical coordinate system defined in Figure 3.1.

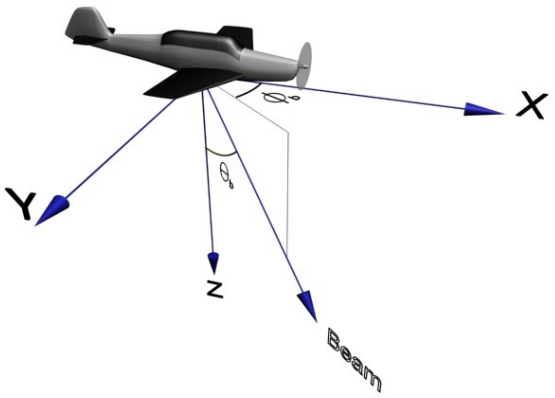

Figure 3.1 UAV centered frame of reference.

A simulation model is constructed using MATLAB [28], which will calculate the radiation pattern of the array based on the array geometry and the beam forming algorithm. The model and design of the basic printed element with the proposed array geometry (a printed rectangular patch) were simulated using HFSS [29]. HFSS is a full wave electromagnetic field solver that is used in industry and academia to design and model antenna structures. The results of the designed and simulated single patch were forwarded to the MATLAB [28] code in order to get the total radiation pattern of the designed array according to equation (3.1) [30]:

$$Gain_{total\ dB}(\theta,\phi) = AF_{dB}(\theta,\phi) + Gain_{single\ patch\ dB}(\theta,\phi) \tag{3.1}$$

Where $AF_{dB}(\theta,\phi)$ is the array factor of the proposed geometrical structure as a function of θ and ϕ in dB. The array factor depends on the geometrical construction of the antenna array and is given by equations (3.2), (3.3), and (3.4) [30]:

$$AF(\theta,\phi) = \left(\frac{sin\left(\frac{M}{2}\psi_x\right)}{sin\left(\frac{\psi_x}{2}\right)}\right)\left(\frac{sin\left(\frac{N}{2}\psi_y\right)}{sin\left(\frac{\psi_y}{2}\right)}\right) \tag{3.2}$$

Where

$$\psi_x = k\, d_x \sin\theta \cos\phi + \beta_x \tag{3.3}$$

$$\psi_y = k\, d_y \sin\theta \sin\phi + \beta_y \tag{3.4}$$

Where k is the wave number, M and N are the number of elements in the x and y directions, respectively, d_x and d_y are the inter-element distances in the x and y directions, respectively, and β_x and β_y are the phase differences between neighboring elements in the x and y directions respectively. The beam steering algorithm determines the steering angles θ_b and ϕ_b and accordingly the phase differences ($\beta_x\ and\ \beta_y$) are calculated using equations (3.5) and (3.6). Note that equation (3.1) covers the complete (θ,ϕ) plane for a certain θ_b and ϕ_b.

Table 3.1 Design parameters for a 2x7 antenna array

Parameter	Value	Units
Antenna parameters		
Substrate width	66	mm
Substrate length	58	mm
Patch width	36	mm
Patch length	28	mm
Substrate ε_r	4.4	unit less
Thickness of metal layer	36	µm
Thickness of substrate	0.8	mm
inter element spacing dx	30	mm
inter element spacing dy	30	mm
Frequency	2.45	GHz
Mini-telemaster UAV [1]		
Usable single wing length	500	mm
Usable single wing width	155	mm
Usable Fuselage length	650	mm
XBee Pro Module [2]		
Transmit power	10	dBm
Receiver sensitivity	-102	dBm

The designed antenna array took into account the size of the wing of the mini-Telemaster UAV. A thin commercial FR-4 material was used. All parameters used in the design and simulations are reported in

Table 3.1 for the antenna array, the UAV, and the transceiver.

Figure 3.2 shows the rectangular structure of the planar 2×7 printed antenna array of patch elements. The beam is steered in a certain (θ_b,ϕ_b) direction by changing the progressive phase shifts according to equations (3.5) and (3.6) [30].

$$\beta_x = -k\, d_x \sin\theta_b \cos\phi_b \tag{3.5}$$

$$\beta_y = -k\, d_y \sin\theta_b \sin\phi_b \tag{3.6}$$

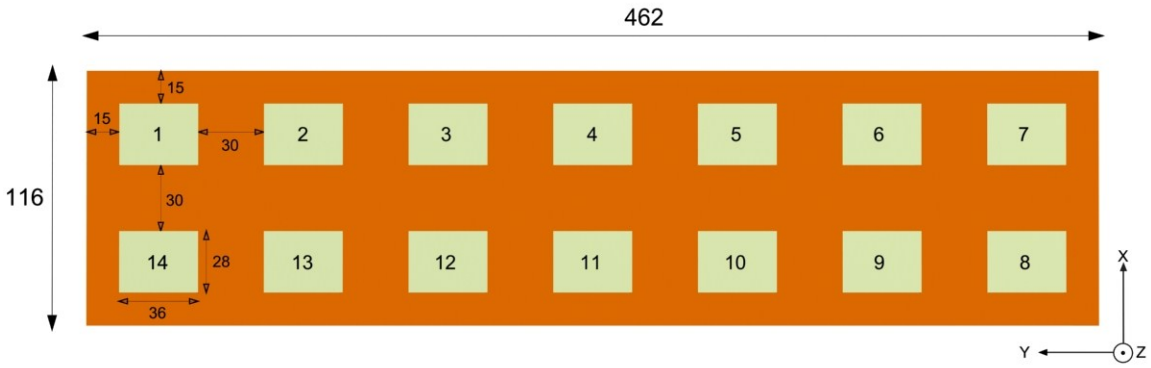

Figure 3.2 Designed planar 2×7 antenna array (all dimensions are in mm).

By evaluating the increase in the antenna gain, we will be able to calculate the increase in the communication range given the sensitivity level, and the transmit power of the XBee module [22] to be used at the ground station and onboard the UAV.

3.4 Simulations

In this section we will present the simulations performed to characterize the antenna array and explore its properties. For any antenna array there must be a feed network with enough output ports to feed all the elements in the array. Due to the limited number of

output ports in RF feed networks that are currently available[3], we reduced the number of elements in our M×N array from 14 (2×7) to 12 (2×6) in the final hardware prototype. In this section we will show the analysis and characterization of both arrays; the 14 elements and the 12 elements.

3.4.1 The 14 elements array

In this work a single patch antenna was designed and modeled using HFSS [29], this patch element is shown in Figure 3.3 where its 3D gain pattern is shown in Figure 3.4. The maximum gain produced from this antenna was 2.5 dBi. The gain pattern as a function of θ and ϕ was saved in a file and called from the MATLAB code to process the array performance. This modeling method will yield better results than using the approximate formulas for patch antennas based on infinite ground plane structures that are usually given in antenna text books.

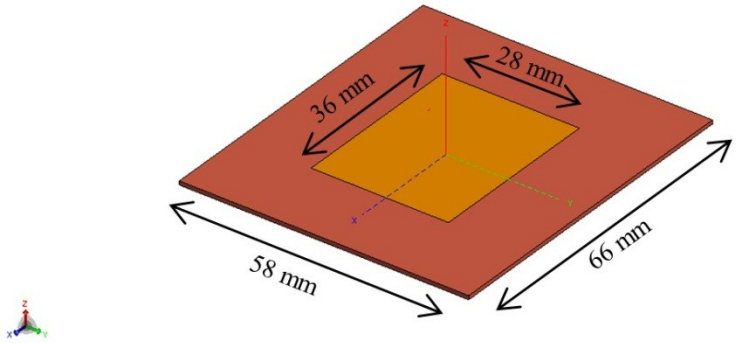

Figure 3.3 The single patch created in the electromagnetic simulation tool (HFSS [29])

[3] Another research group member has developed a 12 port power combiner RF circuit that will be used in combining signals from different elements. Thus, we had to reduce the number of antenna elements in the planar array from 14 to 12.

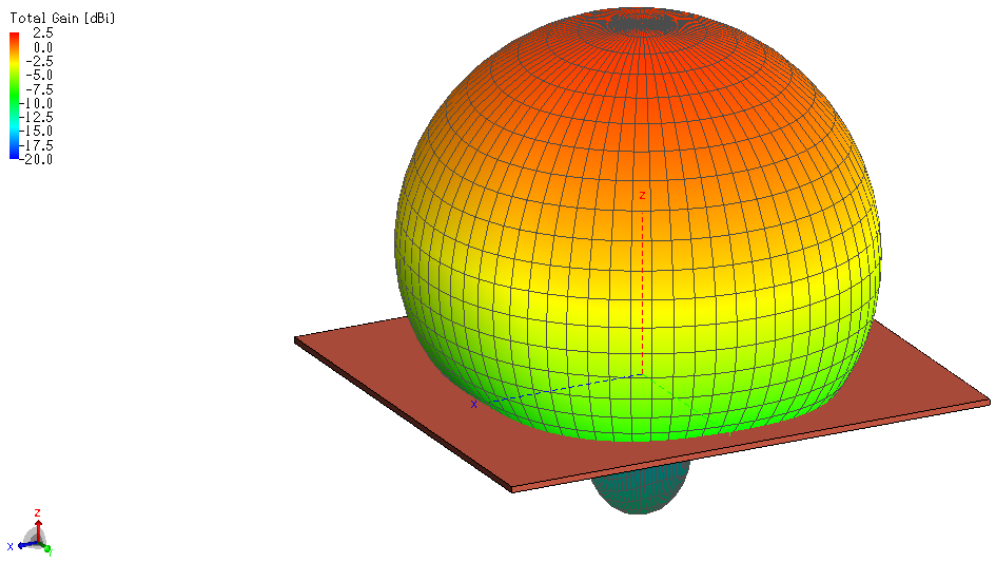

Figure 3.4 Radiation pattern of the single patch element at 2.45 GHz.

To check the resonant frequency of the patch element, the return loss (S_{11}) is plotted in the frequency domain. This is shown in Figure 3.5 where the minimum return loss is almost at 2.45 GHz indicating that the patch will resonate at that frequency, where the $-10\ dB$ bandwidth is 50 MHz.

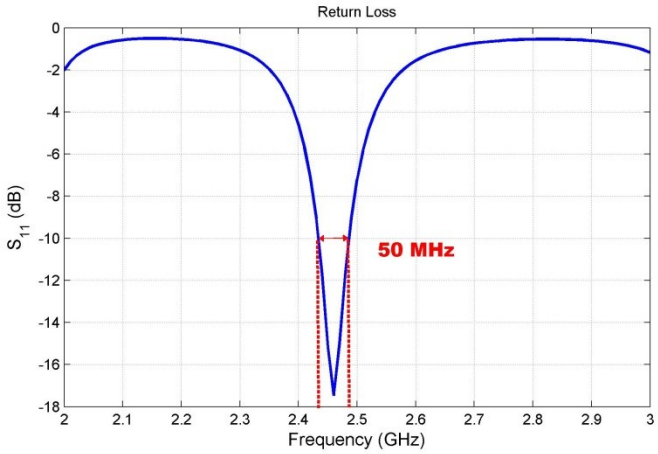

Figure 3.5 Simulated return Loss (S_{11}) Curve.

The maximum number of patch antenna elements to be placed within a single wing of a mini-Telemaster UAV platform [23] is 14, arranged in a 2×7 grids as shown in

Figure 3.2. This is due to the limited wing space within this specific UAV model. The resulting 3D gain pattern of the 2×7 planar array is shown in Figure 3.6 and Figure 3.7. The array gain pattern was obtained using the array pattern multiplication principle stated in equation (3.1). It is evident that at the given steering angles ($\theta_b = 0°, \phi_b = 0°$) the beam is narrow in the y direction (at $\phi = 90°$) where the half power beam width (HPBW) is 16°, while it is wide in the x direction (at $\phi = 0°$) where the HPBW is 49°.

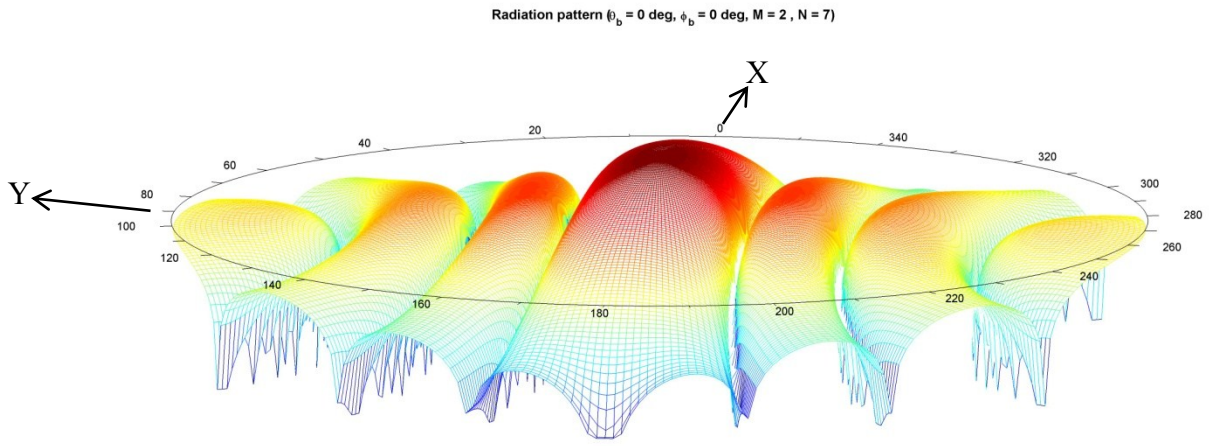

Figure 3.6 3D radiation pattern of the proposed 2×7 antenna array at ($\theta_b = 0°, \phi_b = 0°$), where the circular axis represents ϕ and the radial distance represents θ where at the center $\theta = 0°$, and at the edge $\theta = 180°$.

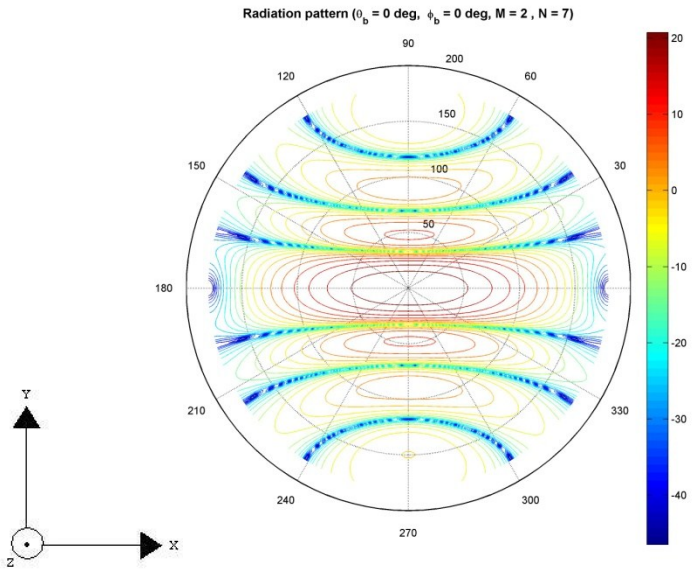

Figure 3.7 A colored contour plot of the radiation pattern (in dB) of the proposed 2x7 antenna array at $(\theta_b = 0°, \phi_b = 0°)$, where the circular axis represents ϕ and the radial distance represents θ where at the center $\theta = 0°$, and at the edge $\theta = 180°$.

To analyze the effect of increasing the number of elements on the maximum gain, the number of elements (N) was increased from 1 to 7 in 1 extra element each time in the y-axis direction (i.e. 2 elements at a time). Figure 3.8 shows the maximum gain values obtained for a beam pointing at $(\theta_b = 0°, \phi = 0°)$, this is the direction at which $(\beta_x \; and \; \beta_y)$ are zeros. It is evident that adding more antenna array elements will give higher gain values. Note that the maximum Gain value is 24.3 dB which is higher than what was produced in [18]. This is due to the difference in the array geometry as well as the number of radiating elements in the array.

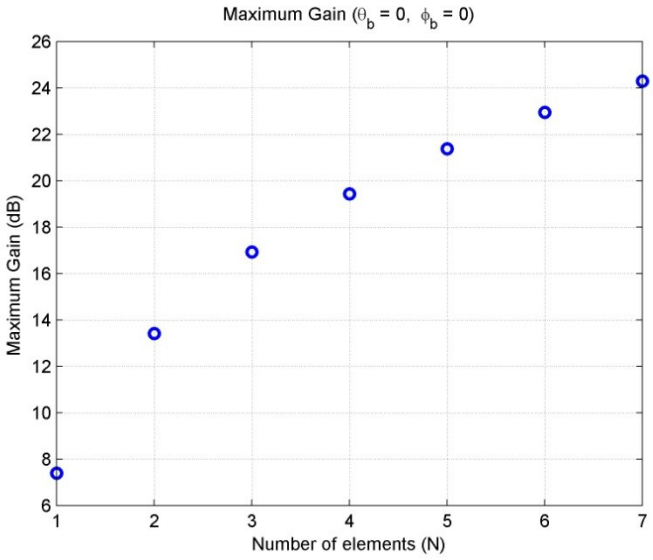

Figure 3.8 Maximum gain produced versus different number of elements in the y direction (N), with M=2.

The gain increase obtained by increasing the antenna elements will directly affect the maximum range of data transfer for this UAV. With a receiver sensitivity of $-102\ dBm$ and a maximum transmitted power P_t of $10\ dBm$ (according to the international value for the rated transmitted power of the XBee module [22]), the maximum line of sight range can be calculated according to the standard path loss model stated in equation (3.7).

$$P_l = 20 * log \frac{4\pi R}{\lambda} \tag{3.7}$$

Where "P_l" is the power losses in dB, "λ" is the wave length in meters and "R" is the line of sight (LOS) distance between the transmitter and the receiver in meters.

First the effective isotropic radiated power (EIRP) must be calculated according to equation (3.8), in which all the variables are resembled in dB except P_t in dBm.

$$EIRP = P_t + G_t - C_t \tag{3.8}$$

Where P_t is the transmitted power, G_t is the gain of the transmitting antenna which is assumed to be $0\ dB$, and C_t is the feeding cables losses in the transmitter which are also assumed to be $0\ dB$. Now to account for the maximum allowable LOS range R we must

consider the minimum allowable received power P_r which is determined by the receiver sensitivity ($-102\ dBm$). Equation (3.9) can be used to calculate the received power P_r.

$$P_r = EIRP - P_l + G_r \tag{3.9}$$

Where G_r is the gain of the receiving antenna and it is substituted by the maximum gain achieved in the generated radiation pattern. This gain is always at ($\theta_b = 0, \phi_b = 0$). This leaves us with only one unknown term in equation 5.5.9 which is a function of the LOS range R. By substituting P_l from equation (3.7) and re-arranging we get equation (3.10).

$$R = \left(\frac{\lambda}{4\pi}\right) * 10^{\left(\frac{EIRP - P_r + G_r}{20}\right)} \tag{3.10}$$

The results are shown in Figure 3.9 where it is clear that the transmission range has increased by almost 6 times when going from 2 elements to 14 elements.

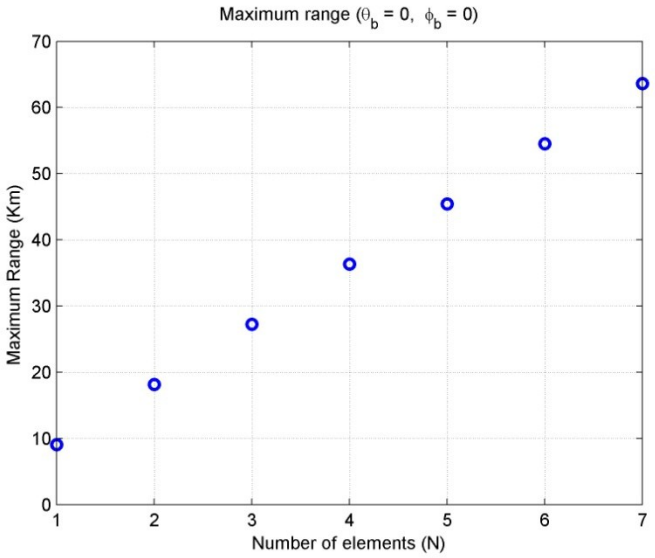

Figure 3.9 Maximum range (Km) Vs. Number of elements (N), where M=2.

Another beam direction is investigated. The beam of the array is directed towards $\theta_b = 45°\ and\ \phi_b = 60°$. The progressive phases (β_x and β_y) of the array elements

calculated according to equations (3.5) and (3.6) are shown in Table 3.2. The 2D radiation gain patterns for this beam direction are shown in Figure 3.10 and Figure 3.11. Figure 3.10 shows the elevation cut (θ cut) when the azimuth angle is fixed at 60°. Figure 3.11 shows the azimuth cut (ϕ cut) when the elevation angle is fixed at 45°, the maximum gain obtained at this direction was $21.5\ dB$. This gives a maximum line of sight (LOS) range of approximately $46.16\ Km$.

Table 3.2 Progressive phases for each element in the array when (θ_b=45°, ϕ_b=60°).

Element	1	2	3	4	5	6	7
Phase	0°	-63.6°	-127.3°	-190.9°	-255.6°	-318.2°	-21.8°
Element	14	13	12	11	10	9	8
Phase	-110.2°	-173.9°	-237.5°	-301.1°	-5.8°	-68.4°	-138.1°

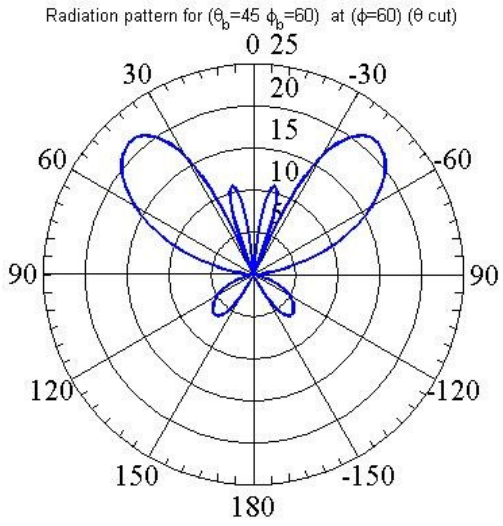

Figure 3.10 Radiation pattern at ($\theta_b = 45°, \phi_b = 60°$) ($\theta$ *cut at* $\phi = 60°$).

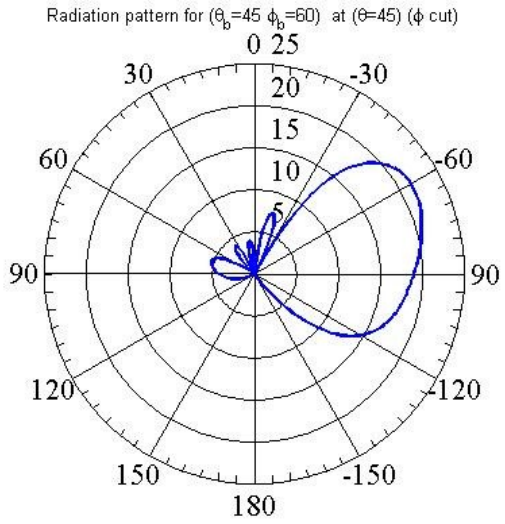

Figure 3.11 Radiation pattern at ($\theta_b = 45°, \phi_b = 60°$) ($\phi$ cut at $\theta = 45°$).

The effect of the number of antenna elements on the half power beam width (HPBW) of the array was investigated. It is clear that the beam width is inversely proportional to the number of elements and this is shown explicitly in Figure 3.12. The HPBW is plotted for the beam direction ($\theta_b = 0°, \phi_b = 0°$) where the beam width is decreased by increasing N, reaching 20° at N=7 in the θ plane. The ϕ plane HPBW is fixed at 45° since the number of elements in the x direction is fixed at 2 elements. This will also be the case for any other beam direction, with a difference in the value of the HPBW. The HPBW of the elevation cut will always be narrower. This is important for our application, as the directed beam from the wing of the UAV is to be as narrow as possible in the elevation plane to focus the radiation pattern towards the ground station.

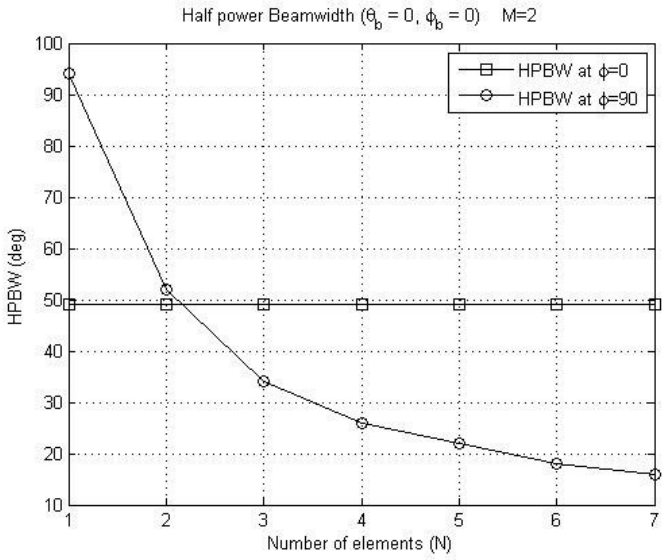

Figure 3.12 HPBW measured from the z-axis at $(\theta = 0, \phi = 0)$ along the θ axis, once at ϕ=0° and once at ϕ=90° Vs number of elements N, where the error margin is ±0.5°.

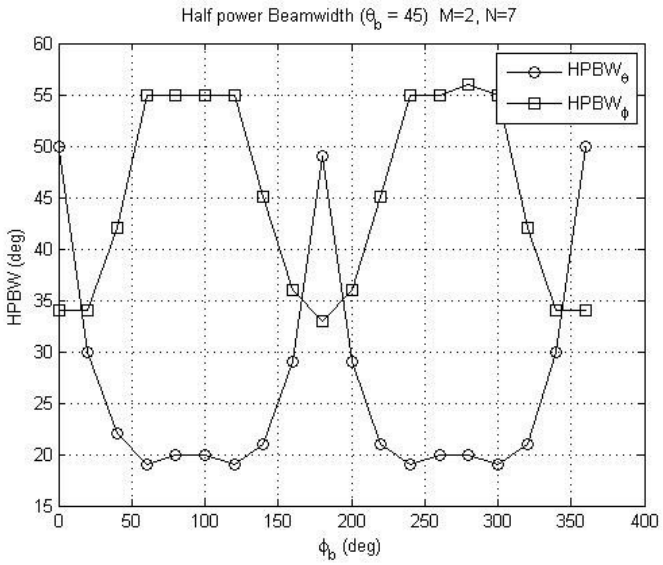

Figure 3.13 HPBW at $\theta_b = 45°$ Vs ϕ_b, where the error margin is ±0.5°.

To investigate the effect of the azimuth angle on the HPBW, θ was fixed and ϕ_b was varied. In Figure 3.13 the beam is scanned along the ϕ axis from 0 to 360 with $\theta_b = 45°$. The error margin in the HPBW results is due to the 1° resolution of gain values generated for the patch element by HFSS [29]. In other words the gain pattern provided by HFSS was generated by scanning the space with a 1° step in both directions (θ *and* ϕ). Notice

that the HPBW values are symmetric around ($\phi_b = 180°$) which is due to the rectangular arrangement of the antenna elements.

To fully characterize the HPBW, a simulation was done to generate the radiation patterns corresponding for all the values of $\theta_b = 0°$ to 180° and for $\phi_b = 0°$ to 360°, keeping in mind that it takes some time to produce one radiation pattern profile for a certain given $\theta_b, and\ \phi_b$, therefore it was done with 5° steps for both beam steering angles in order to have a reasonable running time (nearly 22 hours on an intel core i7 machine running Microsoft Windows 7 and Matlab R2010a).

The produced HPBW values for every given $\theta_b, and\ \phi_b$ is represented in two ways. The first is by calculating the solid angle (in deg^2) bounding the gain values more than the $-3\ dB$ limit, this is shown in Figure 3.14 as a contour plot. The second is by calculating the angular range that includes these gain values, once in the θ direction and once in the ϕ direction, this is shown in Figure 3.15 and Figure 3.16 respectively as contour plots. It is worth mentioning that the minimum HPBW achieved on the θ axis is 14˚ and on the ϕ axis is 25˚.

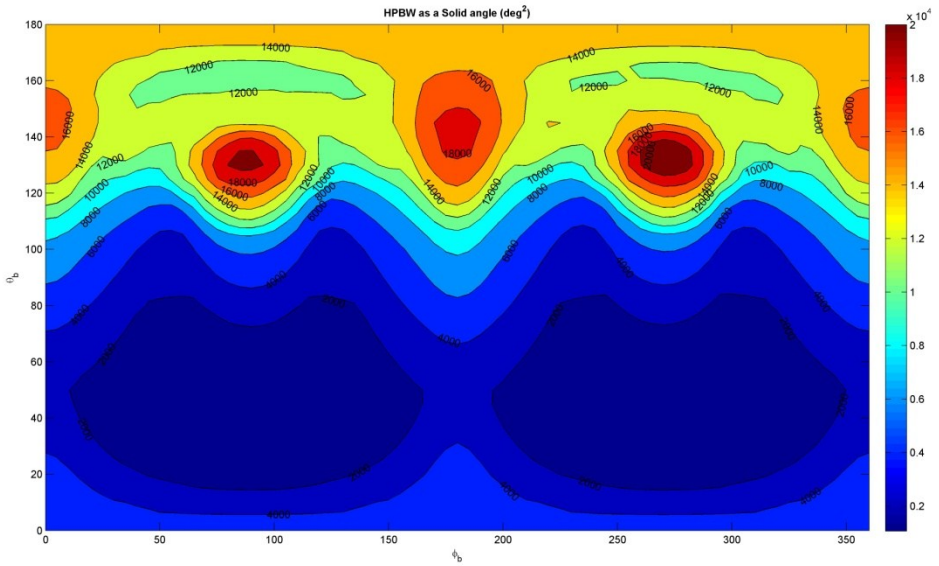

Figure 3.14 A colored contour plot of the HPBW values as a solid angle (deg^2) for $(0^\circ < \theta_b < 180^\circ)$ and $(0^\circ < \phi_b < 360^\circ)$.

The minimum HPBW (θ cut) regions exists for $(50° < \phi_b < 130°)$ and $(230° < \phi_b < 310°)$ on the ϕ_b axis and for $(6° < \theta_b < 45°)$ and $(135° < \theta_b < 175°)$ on the θ_b axis. The minimum HPBW (ϕ cut) regions exists for $(0° < \phi_b < 60°)$, $(110° < \phi_b < 250°)$ and $(290° < \phi_b < 360°)$ on the ϕ_b axis and for $(20° < \theta_b < 160°)$ on the θ_b axis. These regions identify the behavior of the array at different beam directions.

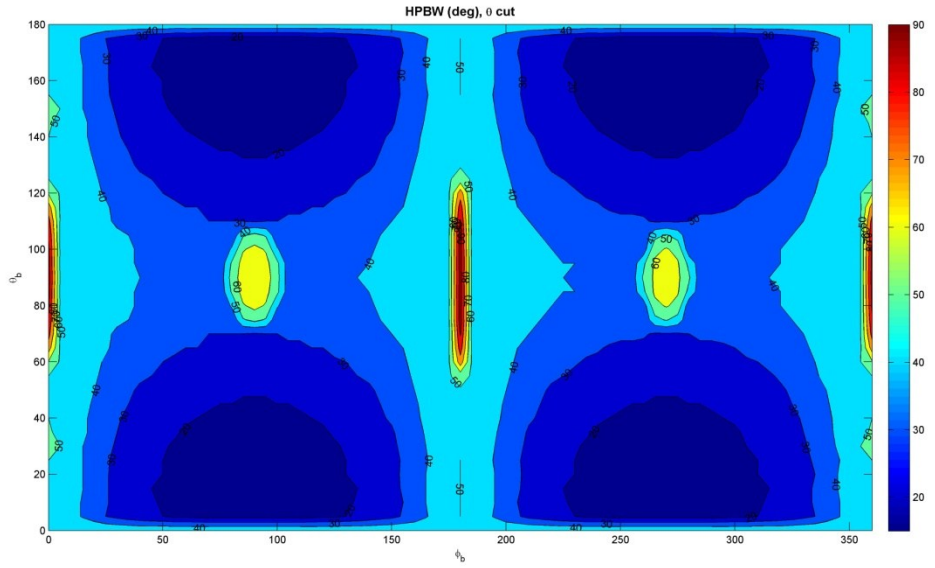

Figure 3.15 A colored contour plot of HPBW as an angular cut (deg) along the θ axis for $(0^\circ < \theta_b < 180^\circ)$ and $(0^\circ < \phi_b < 360^\circ)$.

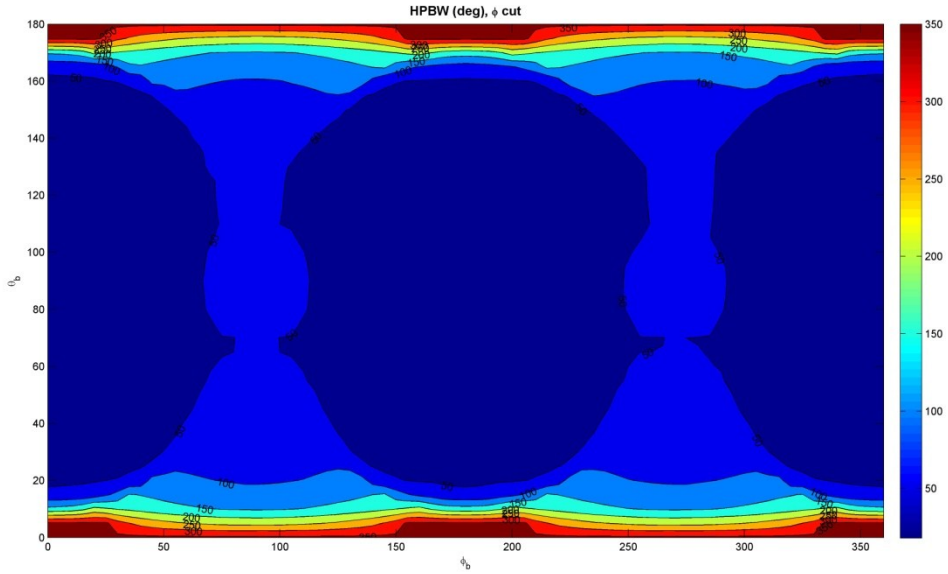

Figure 3.16 A colored contour plot of HPBW as an angular cut (deg) along the ϕ axis for $(0^\circ < \theta_b < 180^\circ)$ and $(0^\circ < \phi_b < 360^\circ)$.

Also to characterize the pointing gain produced at each steering angle, another contour plot is produced. This is shown in Figure 3.17, where the region of maximum pointing

gain is found in $(0° < \phi_b < 360°)$ on ϕ_b axis and $(0° < \theta_b < 17°)$ on θ_b axis. The maximum pointing gain produced is 24.3 dB, thus by using equation (3.10) we can deduce the best LOS operating range achievable to be is 63.65 Km. The region of minimum pointing gain is found in $(120° < \theta_b < 135°)$ on θ_b axis and in $(65° < \phi_b < 110°)$ and $(250° < \phi_b < 290°)$ on ϕ_b axis. The minimum pointing gain is 4 dB, thus by using equation (3.10) we can deduce the worst LOS operating range which is 6.15 Km.

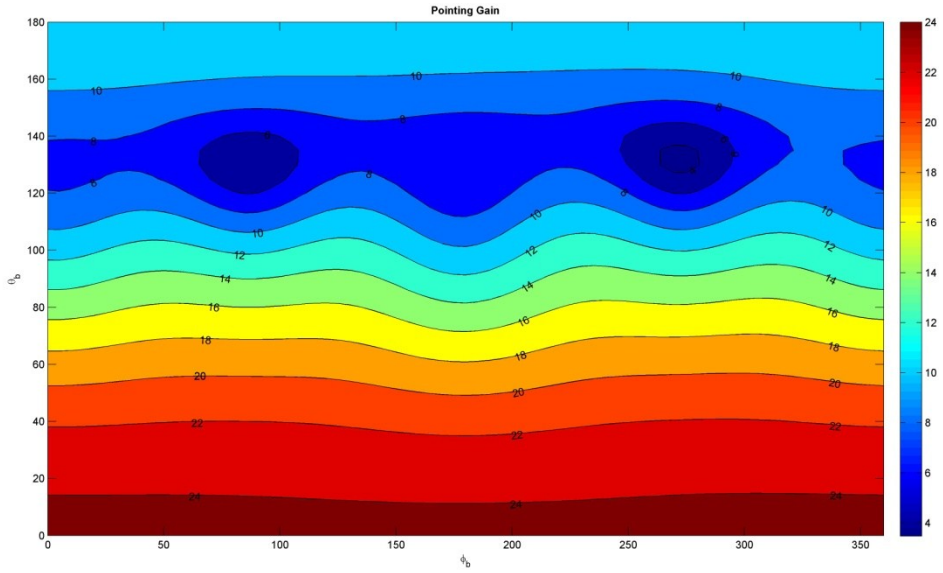

Figure 3.17 A colored contour plot showing the pointing beam gain produced at all possible steering angles (θ_b, ϕ_b).

3.4.2 The 12 elements array

In this section we will present the simulation results for the 12 elements array. These results are generated using the same models for the previous 14 elements array. The only difference is the number of elements N where it is reduced from 7 to 6 due to limitations on the number of the output ports in the available feed network. Thus we have a 2×6 instead of a 2×7 antenna array.

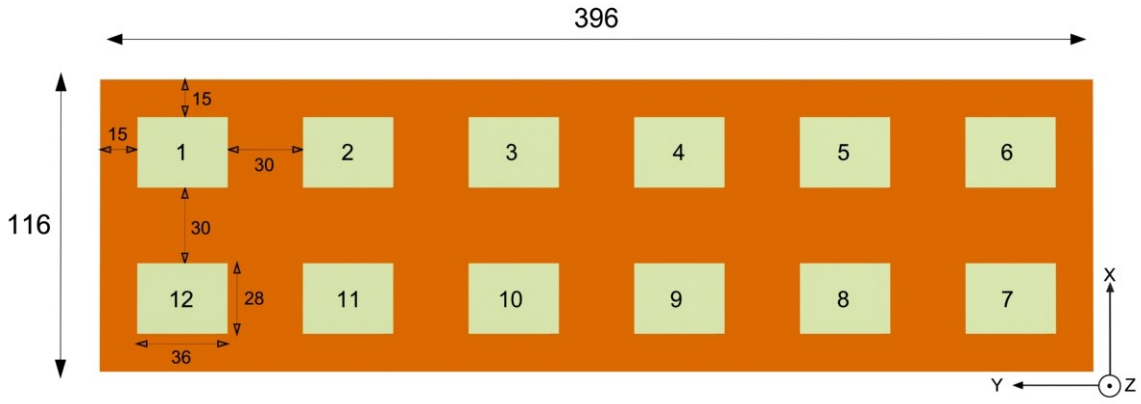

Figure 3.18 Planar 2×6 patch antenna array (all dimensions are in mm).

The 3D gain pattern of the 2×6 planar array is shown in Figure 3.19 and Figure 3.20. The array gain pattern was obtained using the array pattern multiplication principle stated in equation (3.1). It is evident that at the given steering angles ($\theta_b = 0°, \phi_b = 0°$) the beam is narrow in the y direction (at $\phi = 90°$), while it is wide in the x direction (at $\phi = 0°$).

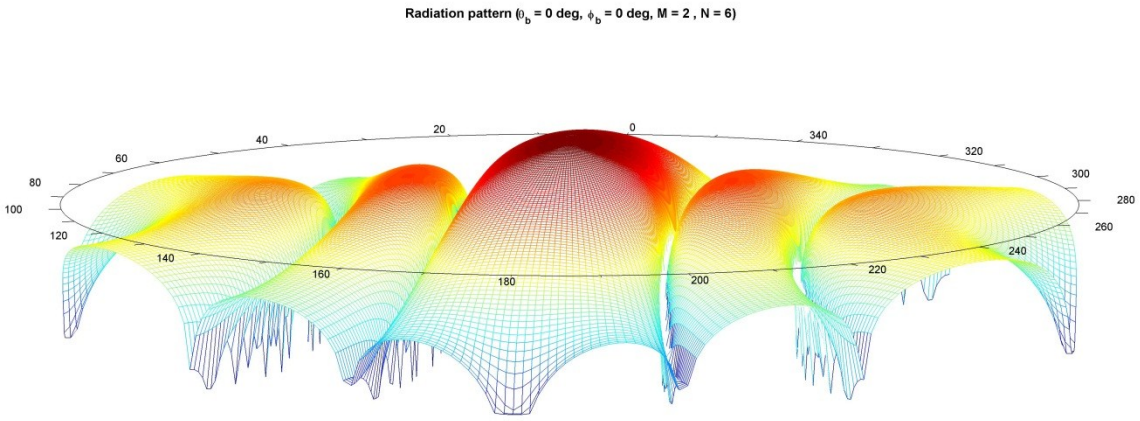

Figure 3.19 3D radiation pattern of the 2×6 antenna array at ($\theta_b = 0, \phi_b = 0$), where the circular axis represents ϕ and the radial distance represents θ where at the center $\theta = 0°$, and at the edge $\theta = 180°$.

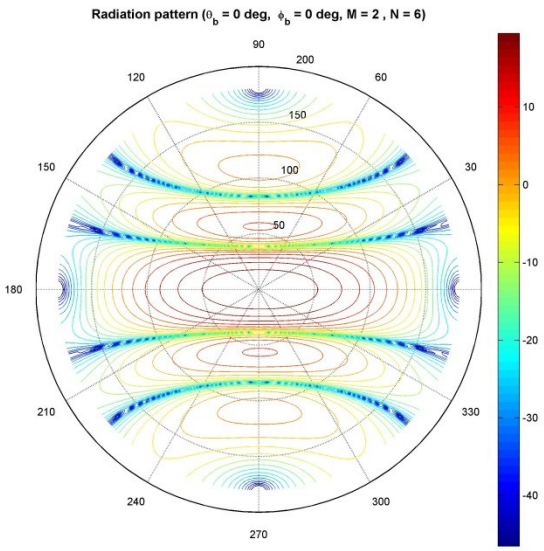

Figure 3.20 A colored contour plot of the radiation pattern (in dB) of the 2x6 antenna array at $(\theta_b = 0°, \phi_b = 0°)$ The circular axis represents ϕ and the radial distance represents θ where at the center $\theta = 0°$, and at the edge $\theta = 180°$.

To fully characterize the HPBW, the simulation performed for the 14 elements array is also performed for the 12 elements array.

Similarly the produced HPBW values for every given $\theta_b, and\ \phi_b$ is represented in two ways. The first is by calculating the solid angle (in deg^2) bounding the gain values more than the $-3\ dB$ limit, this is shown in Figure 3.21 as a contour plot. The second is by calculating the angular range that includes these gain values, once in the θ direction and once in the ϕ direction, this is shown in Figure 3.22 and Figure 3.23 respectively as contour plots. It is worth mentioning that the minimum HPBW achieved on the θ axis is 18˚ and on the ϕ axis is 32˚. Notice how the minimum values increased due to the decreased array size.

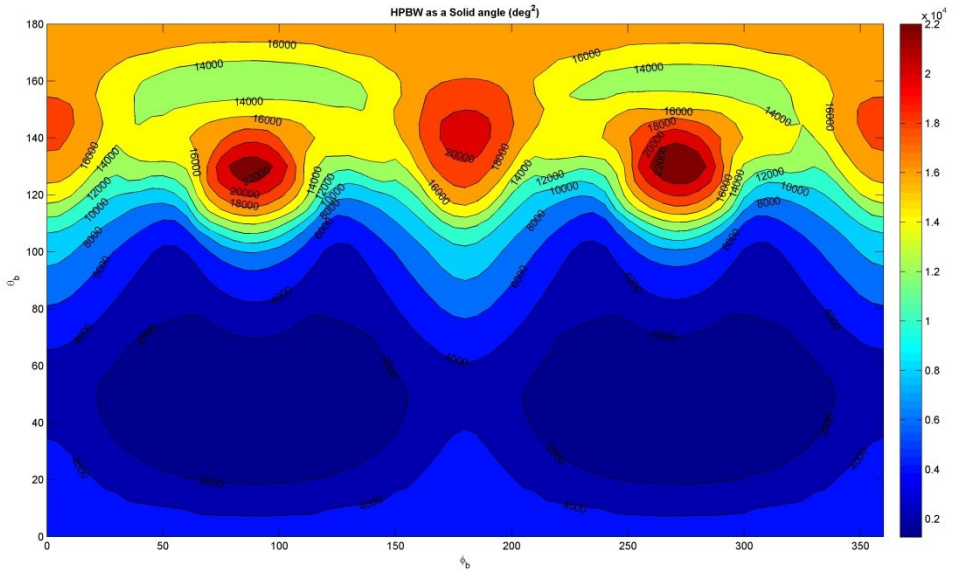

Figure 3.21 A colored contour plot of the HPBW values as a solid angle (deg^2) for ($0^\circ < \theta_b < 180^\circ$) and ($0^\circ < \phi_b < 360^\circ$).

The minimum HPBW (θ cut) regions exists for $(60° < \phi_b < 120°)$ and $(240° < \phi_b < 300°)$ on the ϕ_b axis and for $(5° < \theta_b < 30°)$ and $(150° < \theta_b < 175°)$ on the θ_b axis. The minimum HPBW (ϕ cut) regions exists for $(0° < \phi_b < 70°)$, $(105° < \phi_b < 255°)$ and $(285° < \phi_b < 360°)$ on the ϕ_b axis and for $(20° < \theta_b < 160°)$ on the θ_b axis. These regions identify the behavior of the array at different beam directions.

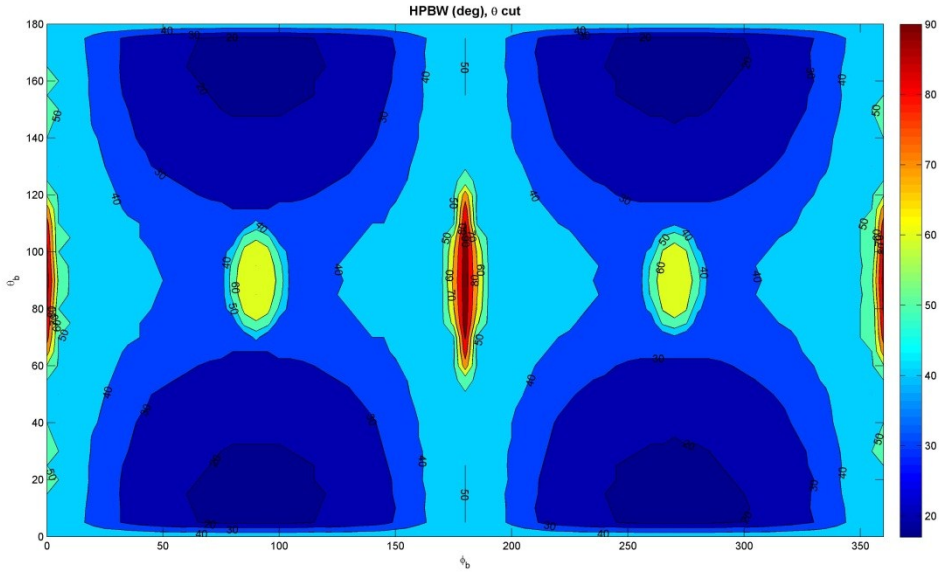

Figure 3.22 A colored contour plot of HPBW as an angular cut (deg) along the θ axis for $(0^{\circ} < \theta_b < 180^{\circ})$ and $(0^{\circ} < \phi_b < 360^{\circ})$.

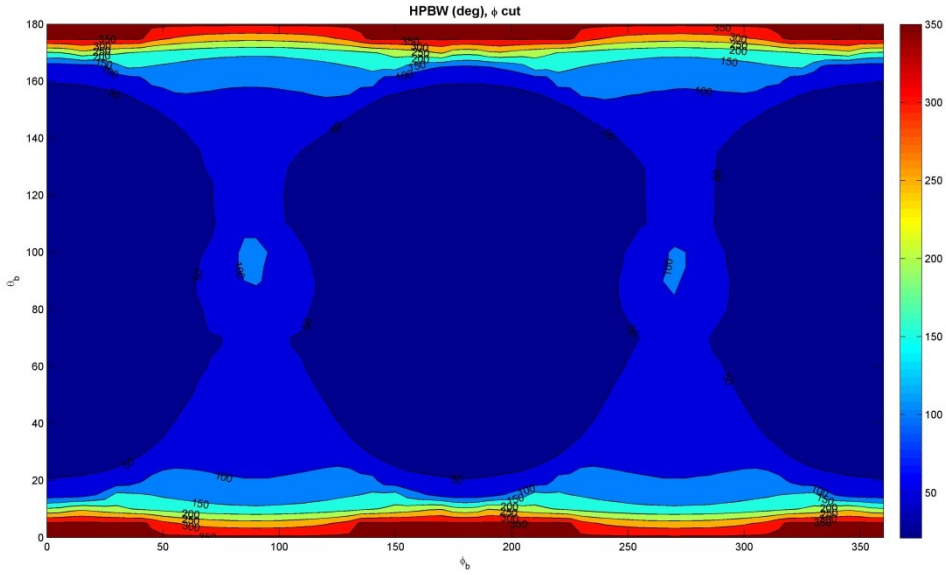

Figure 3.23 A colored contour plot of HPBW as an angular cut (deg) along the ϕ axis for $(0^{\circ} < \theta_b < 180^{\circ})$ and $(0^{\circ} < \phi_b < 360^{\circ})$.

Also to characterize the pointing gain produced at each steering angle, another contour plot is produced. This is shown in Figure 3.24, where the region of maximum pointing

gain is found in $(0° < \phi_b < 360°)$ on ϕ_b axis and $(0° < \theta_b < 22°)$ on θ_b axis. The maximum pointing gain produced is $22\ dB$. Notice that the maximum gain is reduced compared to the 14 element array due to the missing elements. Thus by using equation (3.10) we can deduce the best LOS operating range achievable to be is $48.85\ Km$. Notice the reduced range as well. The region of minimum pointing gain is found in $(120° < \theta_b < 140°)$ on θ_b axis and in $(70° < \phi_b < 100°)$ and $(250° < \phi_b < 290°)$ on ϕ_b axis. The minimum pointing gain is 4 dB, thus by using equation (3.10) we can deduce the worst LOS operating range which is $6.15\ Km$.

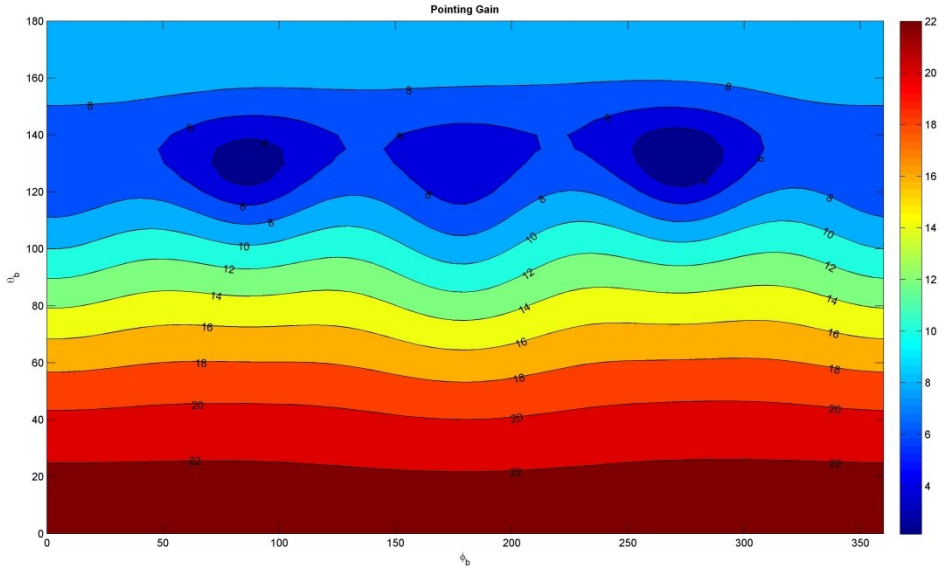

Figure 3.24 A colored contour plot showing the maximum gain produced at each given steering angles (θ_b, ϕ_b).

3.5 Measurements

The 2×7 planar antenna array of patch elements shown in Figure 3.25 and Figure 3.26 was fabricated and measured in the AMSDL (Antennas and microwave structures design lab) in the EE department at King Fahd University for Petroleum and Minerals (KFUPM). Figure 3.25 shows the top view of the fabricated array and Figure 3.26 shows the bottom view where the SMA connectors can be noticed clearly.

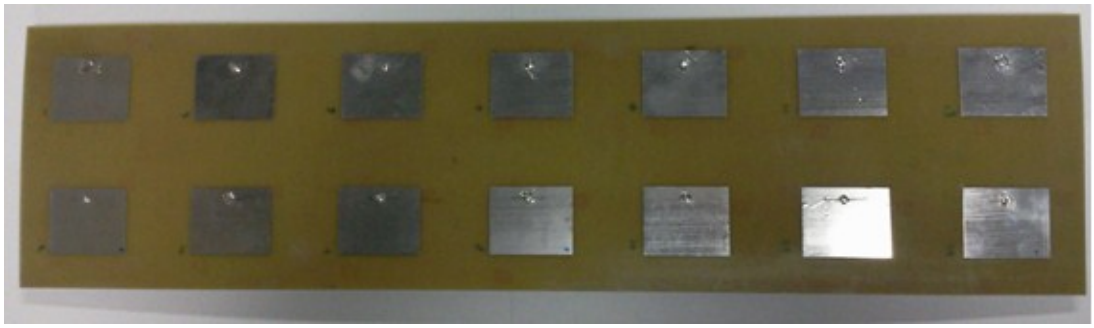

Figure 3.25 The fabricated 2x7 planar antenna array (front side).

Figure 3.26 The fabricated 2x7 planar antenna array (back side).

The inter-element effects of the designed array were characterized and inspected. A series of measurements were made using a vector network analyzer (VNA), the one used in this work is an HP8510C.07.14.

According to Figure 3.2 a number has been assigned to each element in the array, these numbers are then used in the S-parameters characterization in order to match between what was measured in the lab and what is shown in Figure 3.2. Figure 3.27 shows the measured reflection coefficients of the first 6 elements in the array. A minimum $-10\ dB$

band width of 40 MHz was obtained. The measured resonant frequency was 2.48 GHz for all. Starting from Figure 3.28 to Figure 3.33 several mutual coupling curves between adjacent elements are shown. The worst coupling case obtained was $-25\ dB$. This value will not affect the array performance, and using the pattern multiplication principle to get the total array pattern is valid. Note that for every self-reflection parameter measured there is a record in the title showing its $-10\ dB$ bandwidth.

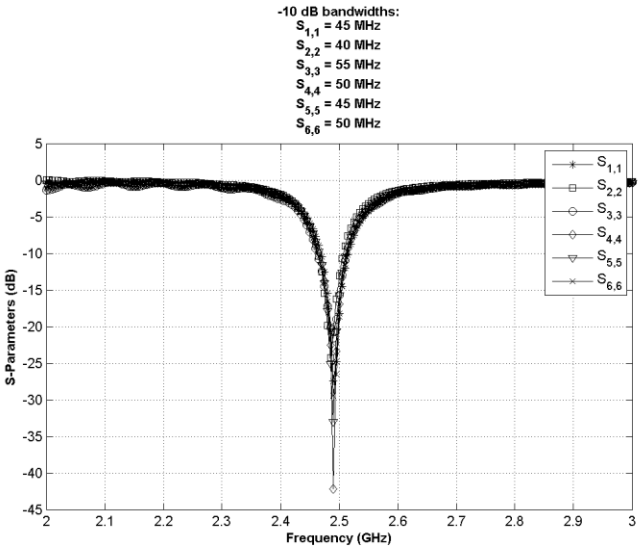

Figure 3.27 Measured reflection coefficient for elements 1, 2, 3, 4, 5, and 6.

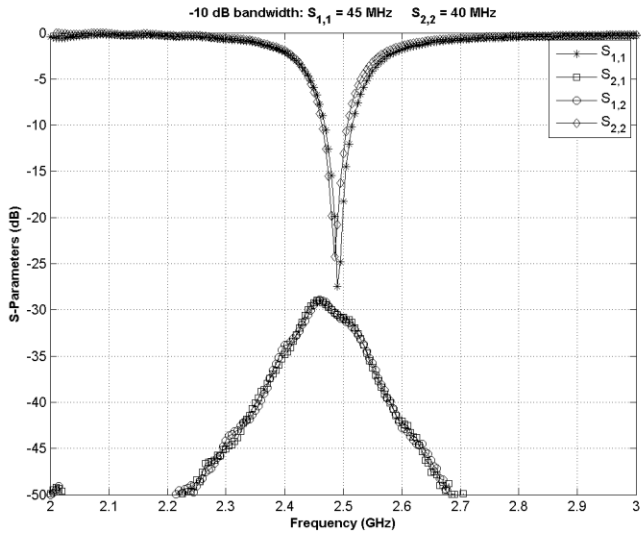

Figure 3.28 Measured reflection coefficient between elements 1, and 2.

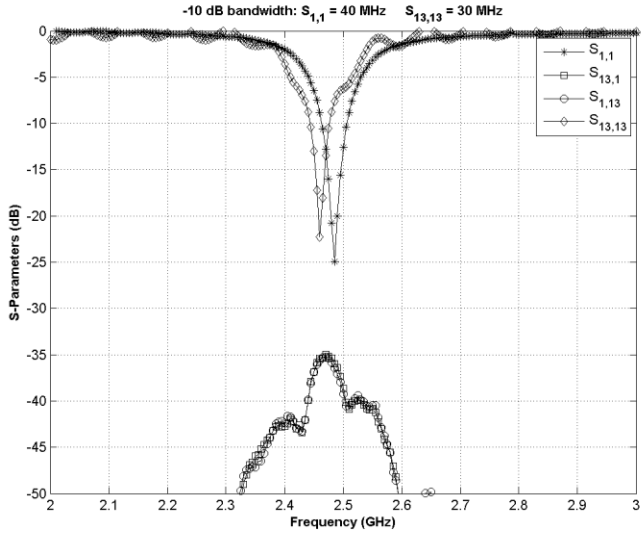

Figure 3.29 Measured reflection coefficient between elements 1, and 13.

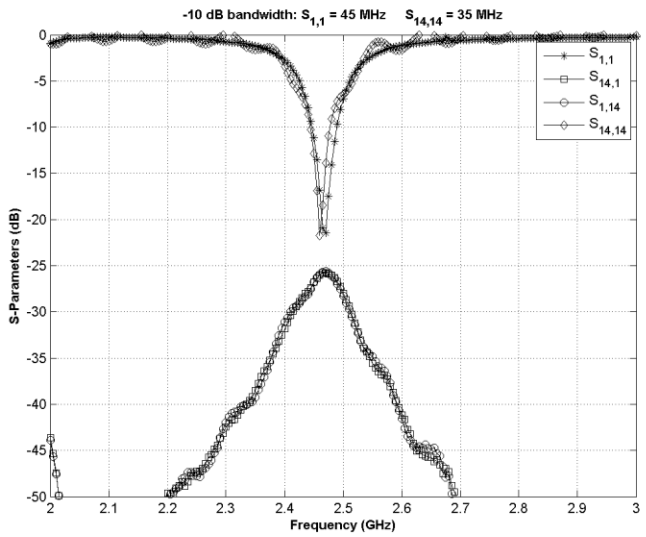

Figure 3.30 Measured reflection coefficient between elements 1, and 14.

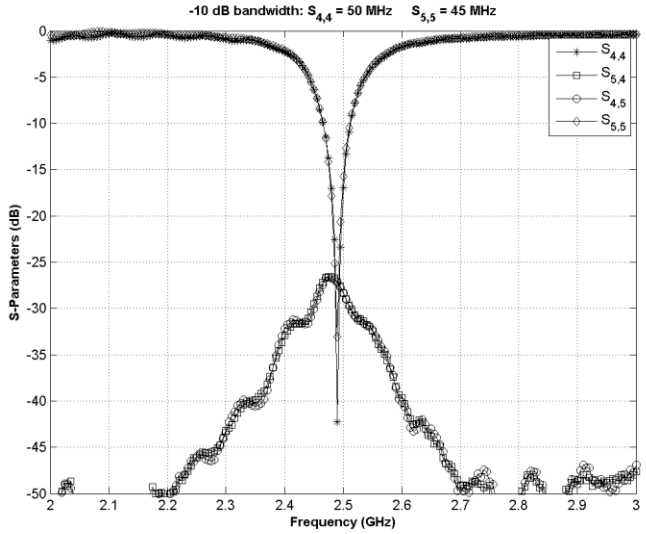

Figure 3.31 Measured reflection coefficient between elements 4, and 5.

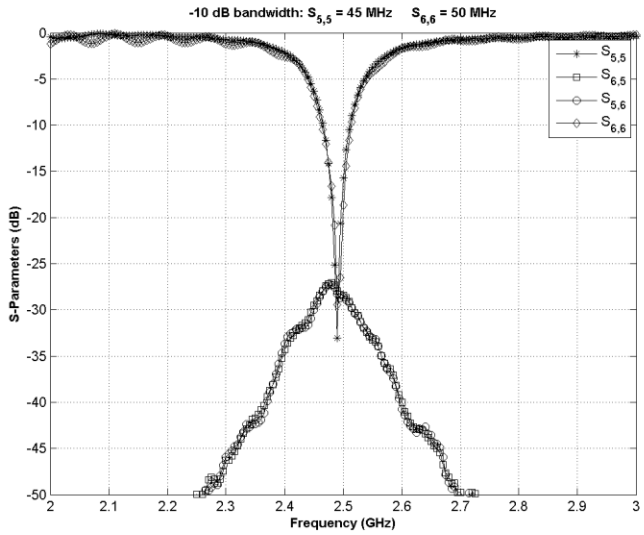

Figure 3.32 Measured reflection coefficient between elements 5, and 6.

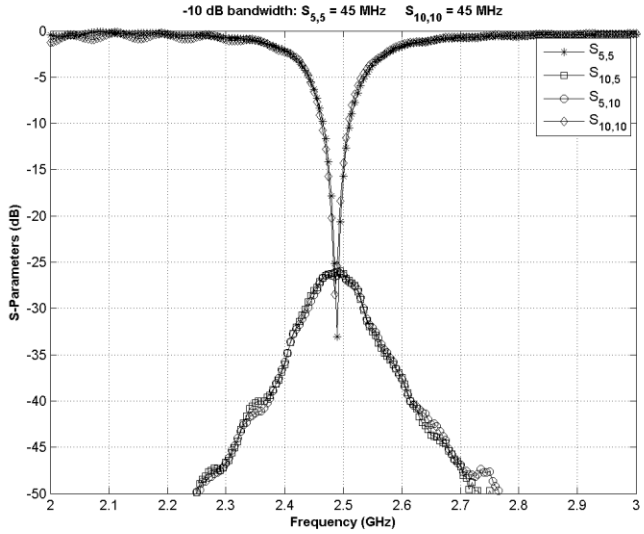

Figure 3.33 Measured reflection coefficient between elements 5, and 10.

The fabricated array is shown in Figure 3.34 after being installed in the wing structure of the mini-Telemaster [23] UAV. Another view of the installation is shown in Figure 3.35 where the back side of the antenna is carrying the subminiature version A (SMA) connectors that will be used to connect the radiating elements to the output ports of the

feed network. The full structure of the mini-Telemaster setup is shown in Figure 3.36 after attaching the antenna to the wing structure.

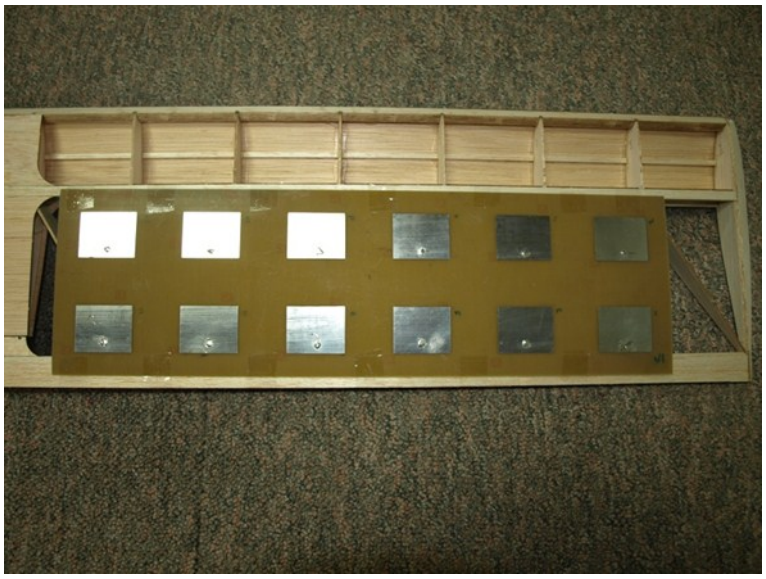

Figure 3.34 The 2x6 array attached to the wing structure of the UAV (downside).

Figure 3.35 The 2x6 array attached to the wing structure of the UAV (upside).

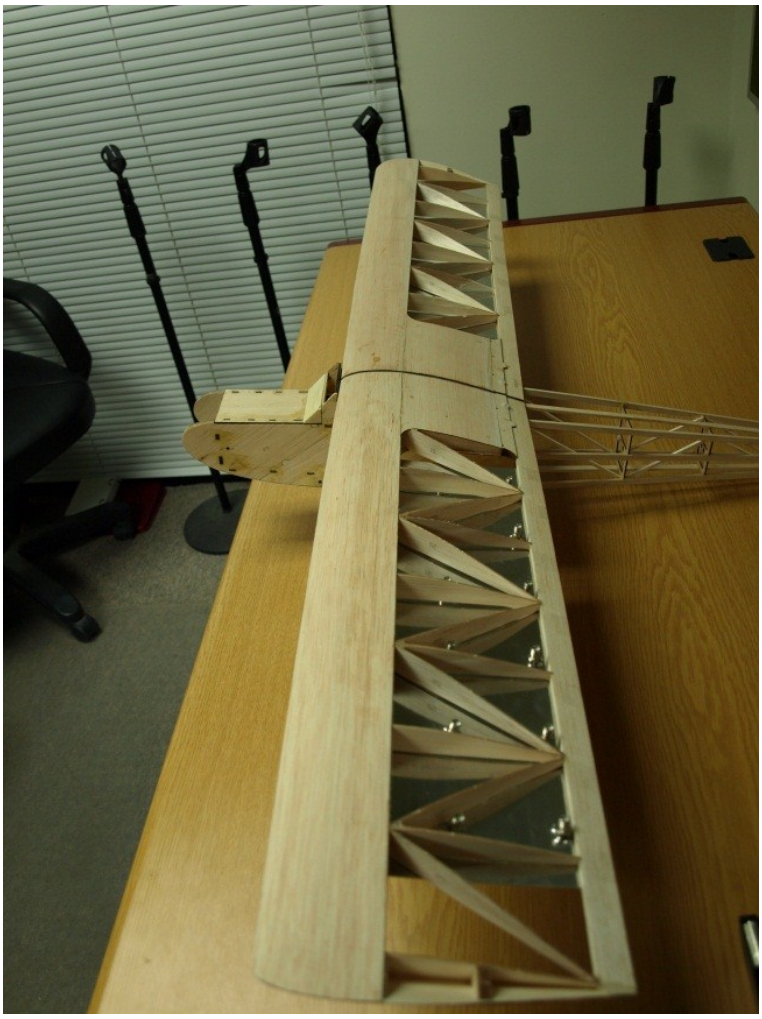

Figure 3.36 The Assembled UAV with the antenna array attached to the wing.

3.6 Conclusions

In this chapter we have presented the design and fabrication of two patch planar antenna arrays, one with 14 elements and the other with 12 elements. We characterized both arrays in terms of their radiation patterns, HPBW and pointing gain versus all possible steering angles (θ_b, ϕ_b).

We also presented the measured S-parameters to quantify the resonant frequency and the isolation between adjacent elements. The antenna array had a resonant frequency of $2.48\,GHz$ with minimum $-10\,dB$ bandwidth of $30\,MHz$ and the adjacent elements

coupling did not exceed $-25\ dB$. We showed also the antenna array mounted inside the wing structure of the mini-Telemaster [23] UAV.

Chapter 4: UAV virtual Trajectory Generator

In order to test the search algorithm that will provide (θ_b, ϕ_b) to the beam forming algorithm to steer the beam, we must have a set of different UAV trajectories. These trajectories could be real vehicle position and orientation data collected from a real flight, or they can be a virtually generated from a computer simulation. In case of collecting real data this requires a ready to fly UAV equipped with a sophisticated inertial measurement unit (IMU) onboard and a well-trained pilot to fly the vehicle in order to collect a lot of trajectory data which will consume a lot of time. Since all of these requirements were not available to us, we decided to go the other way (generating a virtual trajectory).

The virtual trajectory must be guaranteed to follow the same dynamic rules followed by a real vehicle. In order to be sure of this fact before using the trajectory in testing the search algorithm, a feasibility test must be performed on the generated trajectory. This test is performed using a well-known dynamic model [31]. This model is usually used to mathematically describe the motion of any general purpose UAV. The model and the process of using it in testing virtual trajectories are described in this chapter.

4.1 Dynamic model of a UAV

In this section we will introduce the common UAV dynamic model used for describing UAV trajectories in usual cases. According to [32] this model can be applied to this work. This model was also used by [31] in a UAV path planning application. It is described by equations (4.1) to (4.10).

Table 4.1 shows the nomenclature used in the dynamic model.

$$\dot{x} = v \,.\cos(\gamma)\ .cos(\psi) \tag{4.1}$$

$$\dot{y} = v \,.\cos(\gamma)\ .sin(\psi) \tag{4.2}$$

$$\dot{z} = v \,.sin(\gamma) \tag{4.3}$$

$$\dot{v} = \frac{F_T}{M} - g.\sin(\gamma) \tag{4.4}$$

$$\dot{\gamma} = \frac{F_N \,.\cos(\sigma)}{M \,.\, v} - g \,.\frac{cos(\gamma)}{v} \tag{4.5}$$

$$\dot{\psi} = \frac{F_N \,.\sin(\sigma)}{M \,.\, v \,.\cos(\gamma)} \tag{4.6}$$

$$F_T = T \,.\cos(\epsilon) - D \tag{4.7}$$

$$F_N = T \,.\sin(\epsilon) + L \tag{4.8}$$

$$D = \frac{C_D}{2} \,.\rho \,.\, v^2 \tag{4.9}$$

$$L = \frac{C_L}{2} \,.\rho \,.\, v^2 \tag{4.10}$$

Table 4.1 Dynamic model nomenclature

C_D	Coefficient of drag, (a structural constant)	F_T	Resultant force along the velocity vector
C_L	Coefficient of lift, (a structural constant)	F_N	Resultant force normal to the velocity vector
D	Aerodynamic drag force	$\dot{v}$	Acceleration
L	Aerodynamic lift force	$\dot{x}$	Velocity component along x-axis
ρ	Air density	$\dot{y}$	Velocity component along y-axis
v	velocity	$\dot{z}$	Velocity component along z-axis
x	x-coordinate	$\dot{\gamma}$	Pitch rate
y	y-coordinate	ψ	Directional angle (absolute yaw angle)
z	z-coordinate	ϵ	Angle of attack
γ	Pitch angle	M	Vehicle's mass
σ	Bank angle (roll angle)	T	Engine's thrust

The coefficients of drag and lift are vehicular constants that depend greatly on the vehicle's structure. Since our UAV model is not fully characterized dynamically and these coefficients are not recorded in literature for the Mini-Telemaster [23], and since we are only interested in generating a virtual hypothetical trajectory, These coefficients were guessed by looking at a similar manned aircraft. The similarity that will validate our guess must be a structural similarity between the manned aircraft and our UAV model. This guess can be validated by the fact that the drag coefficient C_D and the lift coefficient C_L (given by equations (4.13) and (4.14) respectively) are not a function of the vehicles mass. Thus the difference in mass between the Cessna 172 and the mini-Telemaster will not affect C_D and C_L.

$$C_D = \frac{2F_d}{\rho v^2 A} \tag{4.11}$$

$$C_L = \frac{L}{\frac{1}{2}\rho v^2 A} \tag{4.12}$$

In (4.11) and (4.12), ρ is air density, v is the airspeed, A is the planform area, F_d is the drag force, L is the lift force, C_D is the coefficient of drag, and C_L is the coefficient of lift. Notice that the planform area increases with the size of the aircraft, and so are the drag and lift forces.

After an extensive search for the available similar manned aircrafts, we observed that the most structural similarity is found between the "Cessna 172" which is shown in Figure 4.1 (a) [33] and the mini-Telemaster UAV model in Figure 4.1 (b) [23]. Thus the difference in size between the Cessna 172 and the mini-Telemaster will not have a significant effect on the coefficients of drag and lift.

(a) (b)

Figure 4.1 (a) Cessna 172 manned aircraft, (b) mini-Telemaster UAV model

According to [34] the coefficient of drag C_D of the Cessna 172 is 0.027 and the maximum coefficient of lift C_L (without flaps) is 1.6. These values will be used accordingly in our dynamic model for the mini-Telemaster.

The engine thrust T, banking angle σ, and the angle of attack ϵ are control inputs to the dynamic model. Calculating them to get a desired trajectory is a complex non-linear control problem which is not needed in this work, because we are interested in generating any virtual trajectory constrained only by the UAV model parameters. In other words we don't care about how does the trajectory looks, we just want it to be a realistic UAV trajectory.

In order to judge the feasibility of the trajectory, the control variables (T, σ, ϵ) must be continuous and bounded in magnitude. To compute these variables we must use the dynamic model in reverse by feeding the sample trajectory as a position and attitude samples in time domain. Then for every position and attitude sample we can use the dynamic model to compute the control variables required to obtain the given position and attitude, this is shown in Figure 4.2. Finally, the control variables will be judged by

observing their patterns in search for any discontinuities or unbounded overshoots. If the control variables pass this test, then the trajectory used to generate these control variables will be validated.

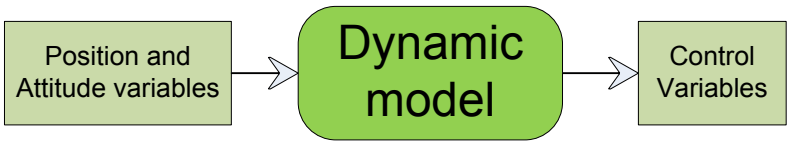

Figure 4.2 Feasibility test process diagram

The next step is to use a virtual trajectory to feed the dynamic model. We initially started by generating a trajectory by ourselves (denoted as the random trajectory generator), then after many trials it didn't produce any feasible trajectories. Thus we went for a readymade flight simulator. We used a flight simulator to fly the vehicle manually and record the trip in order to extract the flight data for further processing.

Section 0 will show the trajectory generator we made as well as some sample trajectories and their feasibility test results. Then section 4.3 will show the trajectory generated by an open source flight simulator and its feasibility test results. The reader can compare easily between the two trajectories and point out the feasible one to be validated for testing our search and tracking algorithms.

4.2 Randomly generated trajectory

This Algorithm generates a series of position and attitude (orientation) samples for a virtual UAV trajectory. The generated trajectory is random every run but it is always constrained by the vehicle specifications stated in Table 4.2 and the limitations stated in Table 4.3.

Table 4.2 Mini-Telemaster specifications [23].

Parameter	Value
Type	Fixed wing
Size	Mini-UAV
Maximum speed	~ 100 km/h (27.77 m/s)
Stall speed	~ 30 km/h (8.33 m/s)
Wing span	1.143 m
Payload	566 gm

Each trajectory sample contains a vector of six real double precision values, three position samples ($X_u\ Y_u\ Z_u$) (Subscript "u" is for UAV) to indicate the vehicle absolute position with reference to the ground station (the signal source) which is assumed to be located at the origin ($X_t = 0, Y_t = 0, Z_t = 0$) (Subscript "$t$" is for ground transmitter), and 3 attitude samples (r, p, y) (r: Roll, p: Pitch, y: Yaw) to indicate the vehicle attitude in 3D space as shown in Figure 4.3.

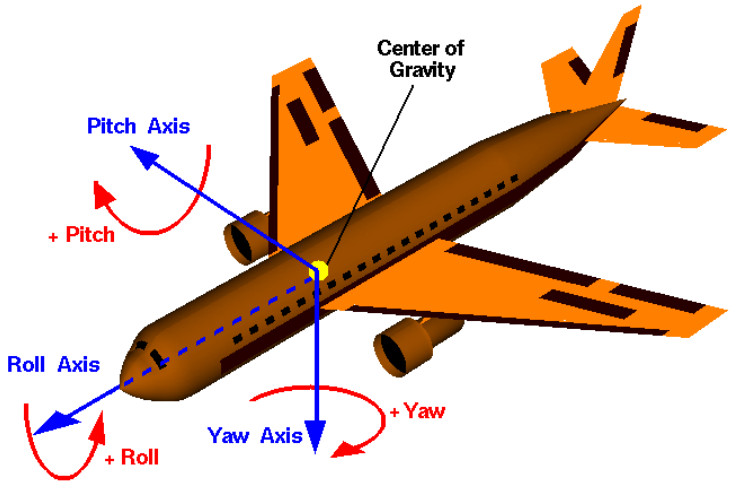

Figure 4.3 a graphical illustration for the attitude angles (roll, pitch and yaw) [7].

This algorithm generates the position samples ($X_u\ Y_u\ Z_u$) first and then it uses them to generate the attitude samples (r, p, y). The time span between each two consecutive trajectory samples is assumed to be 5ms according to the maximum delay between successive readings from the RSSI device that can be used [22]. Figure 4.4 shows the main procedures in this algorithm.

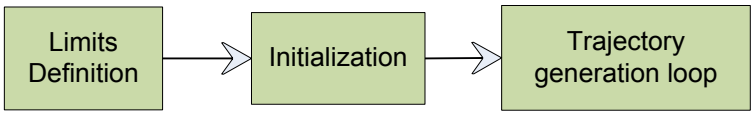

Figure 4.4 Main procedures in the random trajectory generator.

4.2.1 Limits Definition

In this work all turns are assumed to be banked turns in which the vehicle preserves its altitude during the turn. Starting from the speed limits we can define other limits as well. The minimum flying speed is the stall speed "V_s". The stalling speed changes if the plane is performing a banked turn (rolling turn). The accelerated stalling speed "V_{as}" is the new value for the stalling speed according to rolling angle (bank angle) r and the original stall speed value V_s. This relation is stated by equation (4.13) [35].

$$V_{as} = \frac{V_s}{\sqrt{\cos(r)}} \tag{4.13}$$

By defining the stall speed factor as the ratio between V_{as} and V_s and plotting the relation between this factor and the rolling angle r we get the graph in Figure 4.5.

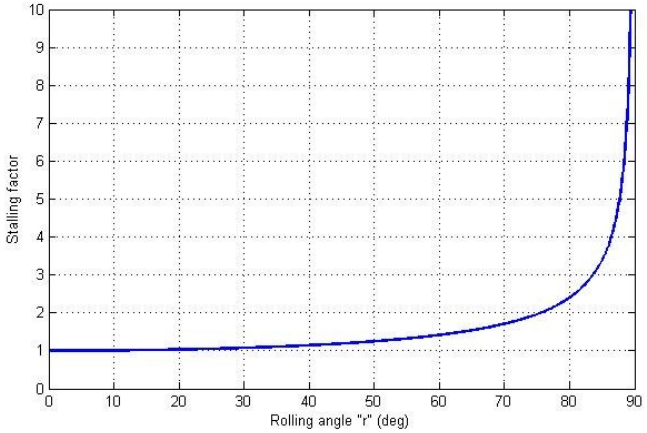

Figure 4.5 Stalling factor versus rolling angle curve.

As shown in Figure 4.5, the stall speed increases as the rolling angle increases. To guarantee a small margin of increase in the stall speed we will consider 50˚ as the maximum possible rolling angle where the stall speed would increase at this value by a factor of 1.247 which is not a significant increase. Now we can define the limits for the rolling angle to be:

$$0° \leq r \leq 50° \tag{4.14}$$

The next step would be to use the limits suggested for "r" in order to deduce the limits for the rate of turn (yaw rate). In [36] a relation between the rolling angle, the yaw rate d_{yaw} and the velocity is used in a certain case study, this relation is given by

$$d_{yaw} = \frac{1091 \tan(r)}{0.51444\, V} \tag{4.15}$$

where V is the vehicle's velocity in m/s, r is the rolling angle in degrees, and d_{yaw} is the rate of turn in deg/sec. It is now obvious that the maximum value for d_{yaw} will be at r_{max} and V_s. By substituting r in equation (4.15) we get equation (4.16) which indicates the dependence of the maximum limit of d_{yaw} upon the current velocity V of the vehicle.

$$\{d_{yaw}\}_{max} = \frac{1091 * \tan(50°)}{0.51444 * V} \tag{4.16}$$

Equation (4.16) will be used only to limit the turning rate when a tight turn is generated in the virtual trajectory. Climbing or falling will not be considered in this simulation while only the cruising part of the trajectory will be simulated for reasons of dynamical model simplification. The typical flying altitude will be maintained throughout the simulation with the addition of some random altitude disturbances.

4.2.2 Initialization

To initialize the algorithm, certain values has to be set at first where later the trajectory generating loop will manipulate them within the limits defined in the previous section. Other quantities will be set and will not change throughout the simulation.

First we begin by generating all assumed variables which are the acceleration "A", the pitch "p", and the yaw rate "d_{yaw}". To generate the acceleration values along the trajectory, the full trajectory will be divided into three equal time segments. During the first segment the vehicle will accelerate from "V_s" to "V_{max}". During the second segment the vehicle will maintain its velocity at "V_{max}" by assuming $A = 0$. Finally during the last segment the vehicle will decelerate from "V_{max}" to "V_s". For the first segment in order to guarantee that the vehicle will reach "V_{max}" at its end, the acceleration value during the first segment will follow equation (4.17). And to assure it reaches "V_s" at the end of the

third segment, the acceleration will take the same value given by equation (4.17), but with a negative sign.

$$A = N\left(\frac{(V_{max} - V_s)}{\frac{st}{3}}, \frac{3 * (V_{max} - V_s)}{\frac{st}{3}}\right) \tag{4.17}$$

"A" is a normally distributed random variable representing the acceleration in m/s^2, "st" is the simulation time in seconds, "V_{max}" and "V_s" are the speed limits of the vehicle. Note that "A" is a normally distributed random variable whose variance is 3 times its mean just to introduce minor disturbances to the velocity curve.

According to the acceleration scenario suggested, the simulation time "st" will be divided into three segments. In the first segment the UAV will increase its velocity from "V_s" to "V_{max}", hence the acceleration required is $\frac{V_{max}-V_s}{\frac{st}{3}}$ which must not exceed the acceleration limit of the vehicle as stated in equation (4.18). In other words the simulation time must obey the condition defined by

$$\frac{V_{max} - V_s}{\frac{st}{3}} \leq A_{max} \tag{4.18}$$

Therefore

$$st \geq \frac{3 * (V_{max} - V_s)}{A_{max}} \tag{4.19}$$

Next we create the pitch "p" along the trajectory, where it is considered as a normally distributed random variable with zero mean and variance "σ_p" defined by equation (4.20).

$$p(k) = N(0, \sigma_p) \qquad \forall\, k \tag{4.20}$$

The altitude disturbances can be controlled by adjusting "σ_p". While the mean value is zero in order to maintain a level flight for the vehicle. The last random variable to be generated is the yaw rate "d_{yaw}", which is created according to equation (4.21).

$$d_{yaw}(k) = N(0, \sigma_y) \qquad \forall\, k \tag{4.21}$$

Where "k" is the sample index, "d_{yaw}" is a normally distributed random variable, and "σ_y" is its variance. Equation (4.21) applies for the whole trajectory except near the middle of the trajectory where we induce a tight turn in the trajectory. This is done by increasing the value of "d_{yaw}" according to a Gaussian function until it reaches its maximum limit stated by equation (4.16) then it starts to decrease by following the Gaussian function until it approaches zero. This is showed later in this section in Figure 4.11.

Table 4.3 shows a list of all the initialized variables with their corresponding initial values in this random trajectory generator.

Table 4.3 list of initialized variables and constant quantities.

Quantity	value	units
Time step between samples "dT"	0.005	s
Simulation time "st"	20	s
V_{min}	$V_s = 8.33$	m/s
V_{max}	27.77	m/s
r_{min}	0°	$degree$
r_{max}	50°	$degree$
σ_p	0.01°	$degree$
σ_y	0.01	deg/s
A_{max}	5	m/s^2
Variable	Initial value	Units
X_u	0	m
Y_u	0	m
Z_u (altitude)	100	m

4.2.3 Trajectory generation loop

After generating all random variables, the algorithm will begin generating the position samples (X_u, Y_u, Z_u). First Z_u will be maintained at a certain value with the effect of some disturbances coming from pitch disturbances. Then each sample of X_u and Y_u will be generated using the previous two samples of X_u and Y_u. In Figure 4.6 we can see a top view of three consecutive samples in the trajectory.

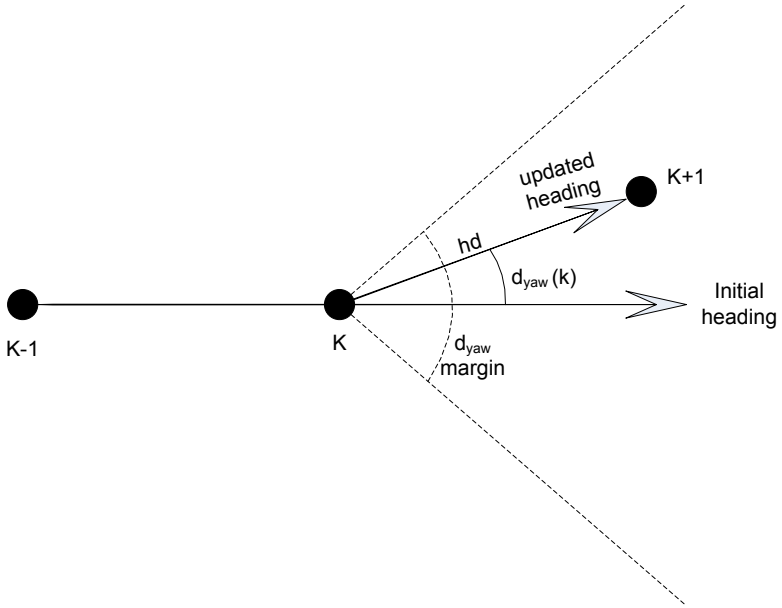

Figure 4.6 a top view of 3 consecutive trajectory samples showing how d_{yaw} is updated, hd: horizontal displacement.

Every loop run will generate a new sample by varying the previous sample within the predefined limits. At the sample k, the new value for the yaw will follow equation (4.22).

$$y(k) = y(k-1) + d_{yaw}(k) \tag{4.22}$$

Where the yaw "y" is the angle measured from the $+ve$ X-axis to the vehicle's heading. After updating the yaw value we must update the horizontal displacement value "hd", but also within the predefined limits of velocity and acceleration. After assigning the acceleration value to a certain sample, it is easy to update its velocity then its horizontal displacement value "hd" using equations (4.23) and (4.24) respectively.

$$V(k) = V(k-1) + A(k) * dT \tag{4.23}$$

$$hd(k) = V(k) * dT \tag{4.24}$$

Where $V(k)$ is the vehicle's velocity at the sample k, $A(k)$ is the acceleration and "dT" is the time interval between the two samples k and $k-1$. This time interval is set to $5\ ms$, which is the typical time delay between two consecutive RSS readings taken by the XBee receiver module [22].

The new horizontal coordinates of the sample "k" can be easily deduced using simple trigonometric relations given by

$$x_u(k+1) = x_u(k) + hd(k) * \cos y(k) \tag{4.25}$$

$$y_u(k+1) = y_u(k) + hd(k) * \sin y(k) \tag{4.26}$$

Calculating the rolling angle r depends on the value of d_{yaw} in degrees per second and vehicle's velocity V according to equation (4.27) which is a rearrangement of equation (4.15). Note that the d_{yaw} direction is opposite to the roll direction defined in Figure 4.3, and this the reason for the $-ve$ sign in equation (4.27).

$$r(k) = -\tan^{-1}\left(\frac{d_{yaw}(k) * 0.51444 * V(k)}{1091}\right) \tag{4.27}$$

The final value to be calculated is Z_u which is easily calculated using a simple trigonometric approach. According to Figure 4.7 the value of $Z_u(k+1)$ can be calculated using

$$Z_u(k+1) = Z_u(k) + hd(\mathrm{k}) * \tan p(k) \tag{4.28}$$

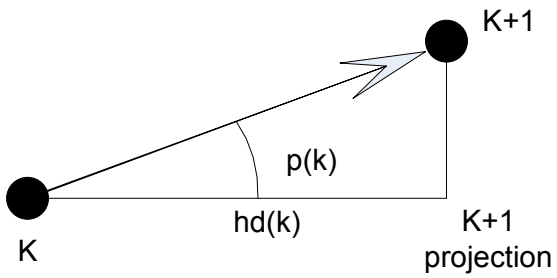

Figure 4.7 a side view for two consecutive trajectory samples.

where $p(k)$ is the pitch value at time k, $hd(k)$ is the horizontal displacement at time "k". Notice that "$p(k)$" is positive or negative, which in turn will induce altitude increments or decrements respectively. Throughout the simulation time, the pitch "p" is a normally distributed random variable with a zero mean and 5 degrees variance "σ_p". This is described by equation (4.29), where "dT" is the time step between each two consecutive trajectory samples.

$$p(k) = N(\sin(k * dT), \sigma_p) \tag{4.29}$$

Figure 4.8 shows a clearer picture of how the variables are calculated. Which variables are assumed and which are deduced accordingly. All random variables are circled and all the accordingly calculated variables are squared, provided that all variables are kept within the predefined limits. Note that (X_u, Y_u, Z_u) are the position coordinates of the vehicle's center of mass.

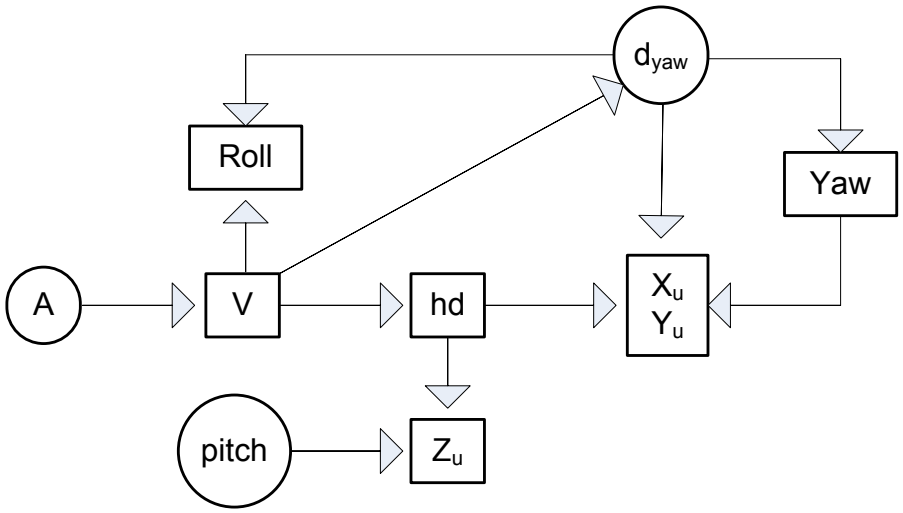

Figure 4.8 Variables flow for a single iteration. Circles are random variables and rectangles are calculated variables.

4.2.4 The Generated trajectory

By running our simulation we generated a sample trajectory which is described in this section. The trajectory is generated to cover a time span of 20 seconds. In Figure 4.9 the generated trajectory is plotted in 3D, where the vehicle started its flight from right above the ground station at ($X_u = 0, Y_u = 0$) and with 100 m altitude. Then it moved preserving its heading and maintaining its altitude until it made a tight banked turn induced by the algorithm in which the vehicle is brought to its maneuvering limits. Finally after the banked turn it retained its heading till the end of the trajectory.

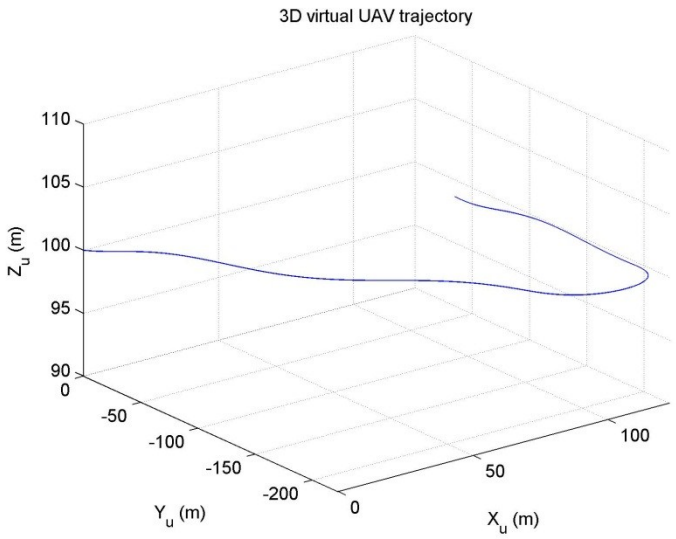

Figure 4.9 3D virtual UAV trajectory

Figure 4.10 is a top view of the 3D trajectory in Figure 4.9, This gives a clear view of the generated trajectory as seen from a reference frame located high above the UAV.

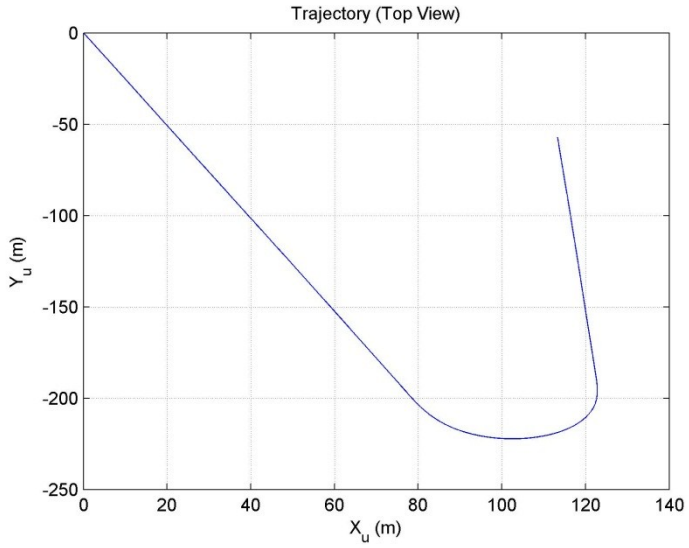

Figure 4.10 A top view of the trajectory shown in Figure 4.9.

Figure 4.11 shows the generated Yaw rate "d_{yaw}" throughout the simulation time (20 seconds). Note that this variable is a normally distributed range limited random variable except in the middle time segment of the trajectory.

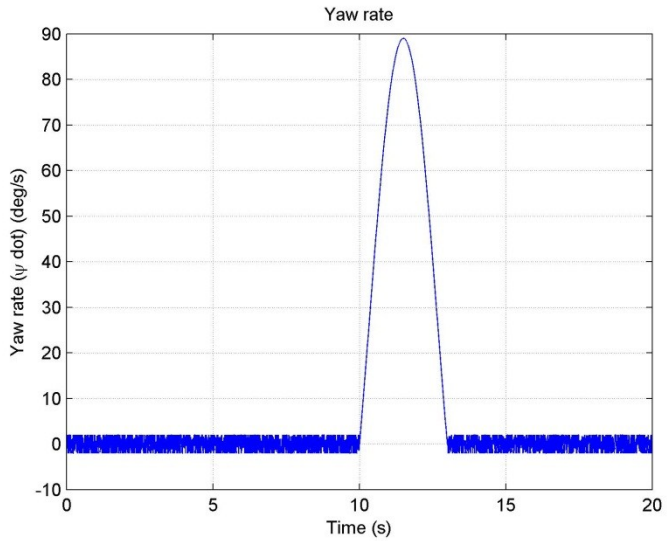

Figure 4.11 Yaw rate (d_{yaw}) versus time.

The vehicle maintained its altitude throughout the simulation. However we induced some disturbances in altitude, these disturbances are shown in Figure 4.12 where we can see that its range is limited within ±1 m.

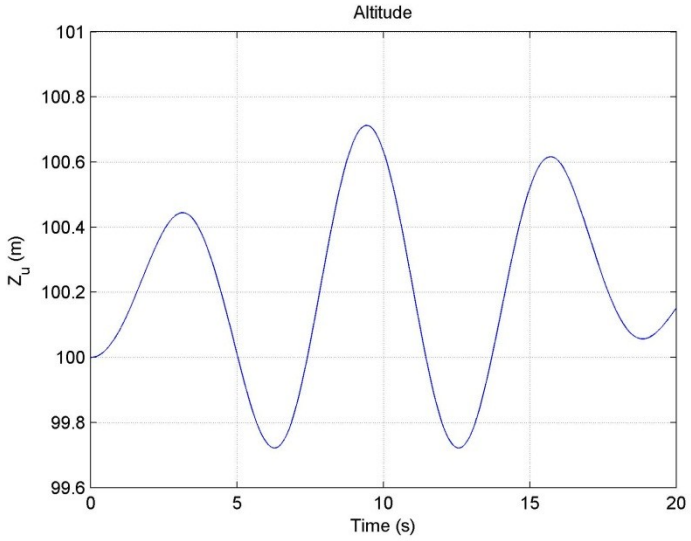

Figure 4.12 Altitude (Z_u) of the trajectory shown in Figure 4.9.

In Figure 4.13 the random variable "p" which represents the pitch angle of the vehicle is shown. Recall that it is a random variable with zero mean and 5 degrees variance.

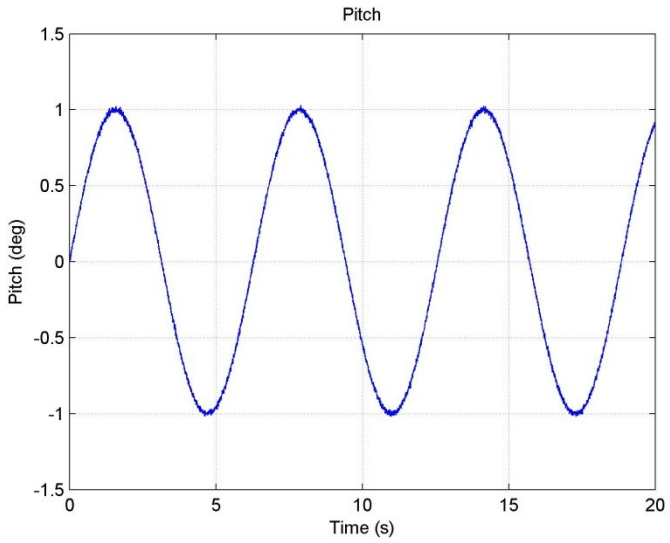

Figure 4.13 Pitch angle (p) of the trajectory shown in Figure 4.9.

In Figure 4.14 the roll angle calculated along the trajectory is shown. Note that there is a rolling pulse in the middle of the trajectory. This indicates the tight turn in the trajectory shown in Figure 4.10.

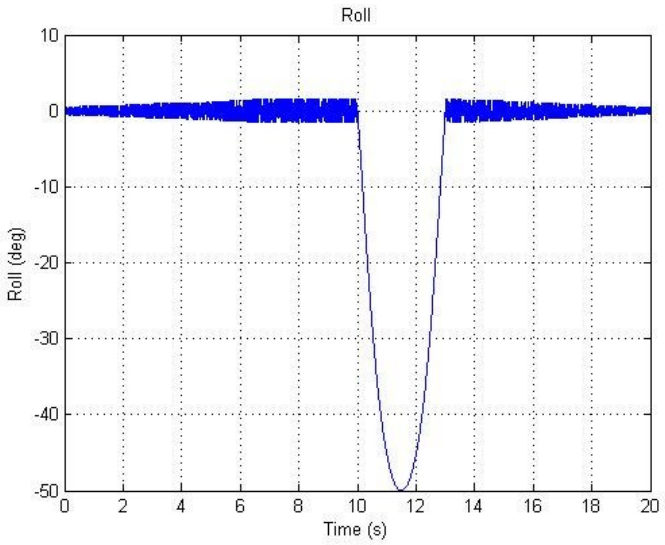

Figure 4.14 Roll angle (r) of the trajectory shown in Figure 4.9.

In Figure 4.15, the yaw angle is shown. Note that the yaw angle is measured from the $+ve$ X-axis to the vehicle's heading. In the beginning the yaw angle maintained a certain

value since the vehicle had a constant heading. In the middle the yaw angle took a step and settled at a new value, this step indicates the tight turn taken by the vehicle. After completing its turn, the vehicle maintained its new heading. This explains the new constant level of the yaw angle which is maintained until the end of the simulation.

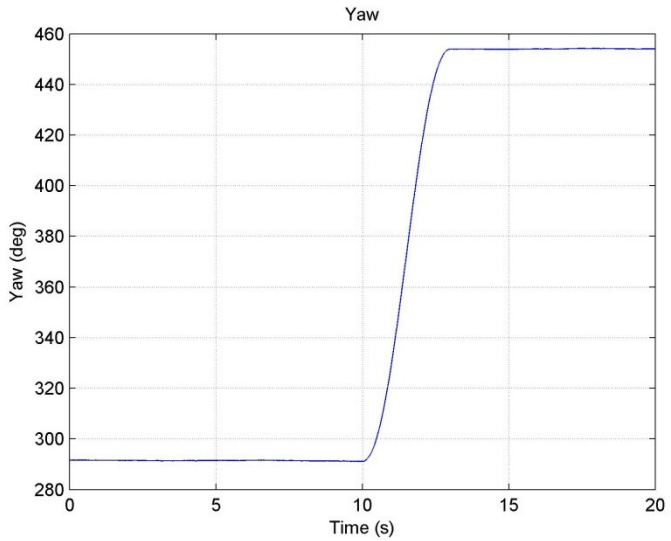

Figure 4.15 Yaw angle (y) of the trajectory shown in Figure 4.9.

Figure 4.16 shows the velocity calculated according to the proposed acceleration scenario. The velocity value increases from its minimum "V_s" to its maximum "V_{max}", then it retains its value for a while and finally it decreases back to its minimum value. Note that the increasing and the decreasing trends contain some fluctuations due to the randomness in the acceleration values.

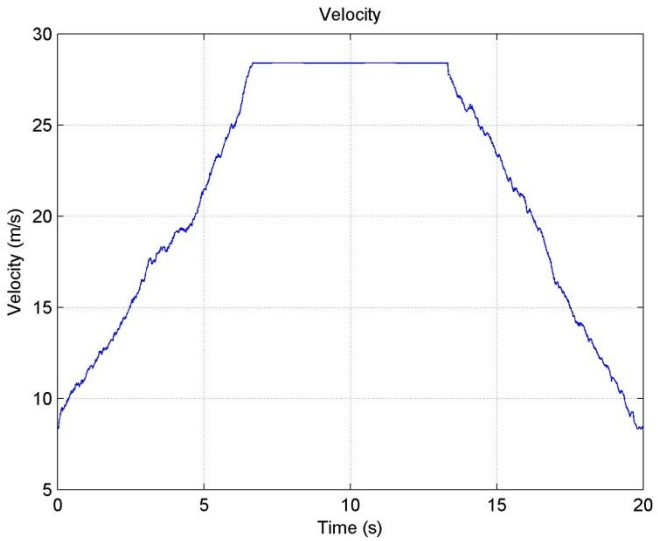

Figure 4.16 Velocity of the vehicle.

4.2.5 Feasibility test

In this section we will conduct a feasibility test on the virtual trajectory generated in the previous section. The feasibility test is carried out in order to validate the trajectory as a trajectory that can be realized by a normal practical engine and a vehicle's control surfaces (flaps, rudder, elevators… etc.).

We conduct this test is by feeding the trajectory to the dynamic model shown before and then calculate the control parameters required to generate the given position and attitude patterns. This is done by reversing the inputs and outputs in the dynamic model stated by equations (4.1) to (4.10), in other words we are working out the model using the block diagram showed in Figure 4.2.

The modified version of the model is stated in the set of equations starting from equation (4.30) and ends by equation (4.47), where the system must run according to the given

order in an iterated loop for all the time samples of the trajectory in order to get the results properly. Notice that this model is not a contribution. It is just another expression of the same dynamic model given in [32].

$$\dot{x}(k) = \frac{x(k+1) - x(k)}{dT(k)} \tag{4.30}$$

$$\dot{y}(k) = \frac{y(k+1) - y(k)}{dT(k)} \tag{4.31}$$

$$\dot{z}(k) = \frac{z(k+1) - z(k)}{dT(k)} \tag{4.32}$$

$$v(k) = \sqrt{\dot{x}(k)^2 + \dot{y}(k)^2 + \dot{z}(k)^2} \tag{4.33}$$

$$\dot{x}(k+1) = \frac{x(k+2) - x(k+1)}{dT(k+1)} \tag{4.34}$$

$$\dot{y}(k+1) = \frac{y(k+2) - y(k+1)}{dT(k+1)} \tag{4.35}$$

$$\dot{z}(k+1) = \frac{z(k+2) - z(k+1)}{dT(k+1)} \tag{4.36}$$

$$v(k+1) = \sqrt{\dot{x}(k+1)^2 + \dot{y}(k+1)^2 + \dot{z}(k+1)^2} \tag{4.37}$$

$$\dot{v}(k) = \frac{v(k+1) - v(k)}{dT(k)} \tag{4.38}$$

$$\dot{\gamma}(k) = \frac{\gamma(k+1) - \gamma(k)}{dT(k)} \tag{4.39}$$

$$\dot{\psi}(k) = \frac{\psi(\dot{k}+1) - \psi(k)}{dT(k)} \tag{4.40}$$

$$\sigma(k) = \tan^{-1}\left(\frac{\dot{\psi}(k)}{\frac{\dot{\gamma}(k)}{\cos(\gamma(k))} + \frac{g}{v(k)}}\right) \tag{4.41}$$

$$F_n(k) = \frac{\dot{\psi}(k) * M * v(k) * \cos(\gamma(k))}{\sin(\sigma(k))} \tag{4.42}$$

$$F_t(k) = M * \left[\dot{v}(k) + g * \sin(\gamma(k))\right] \tag{4.43}$$

$$D(k) = \frac{C_d}{2} * \rho * v(k)^2 \tag{4.44}$$

$$L(k) = \frac{C_l}{2} * \rho * v(k)^2 \tag{4.45}$$

$$\epsilon(k) = \tan^{-1}\left(\frac{F_n(k) - L(k)}{F_t(k) + D(k)}\right) \tag{4.46}$$

$$T(k) = \frac{F_n(k) - L(k)}{\sin(\epsilon(k))} \tag{4.47}$$

Note that k is the sample counter and all the other variables follow the same nomenclature stated in

Table 4.1.

After working out the modified version of the model, the control parameters are shown to evaluate the feasibility of the given trajectory. First the thrust of the vehicle's engine required to produce such trajectory is shown in Figure 4.17. It is obvious that such thrust pattern is unrealizable by any practical engine, where we have a lot of discontinuities in the pattern where the thrust values are fluctuating so frequently in a very small period of time. This will make it impossible for any real engine to follow this pattern successfully.

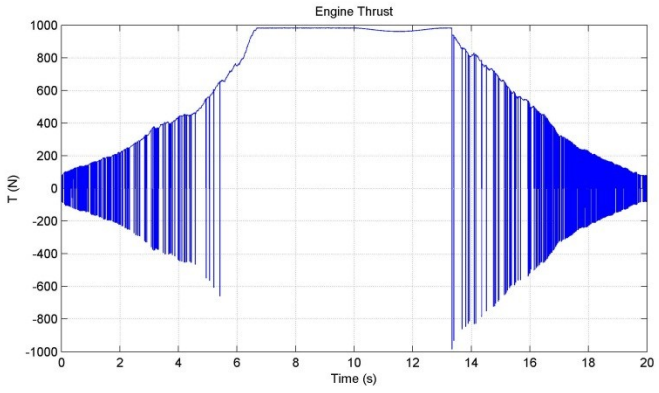

Figure 4.17 Engine thrust.

Then in Figure 4.18 the acceleration pattern is shown. Notice how the pattern is discontinuous except for the region we intentionally assigned a value of zero to the acceleration vector. Also notice the sudden overshoot occurred right after the zero acceleration region. All these observations indicate a nonrealistic behavior of the acceleration.

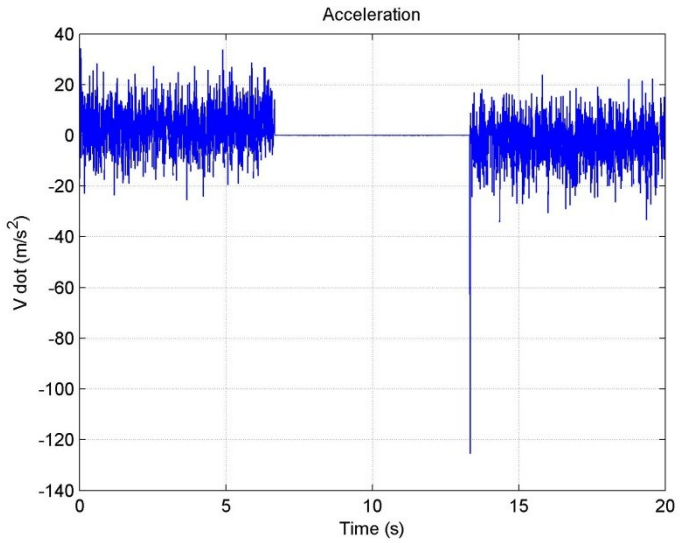

Figure 4.18 Acceleration pattern of the generated trajectory.

Finally, in Figure 4.19 the angle of attack of the vehicle is shown. The same note is taken here where we have a very disturbed pattern with a lot of discontinuities. This will make it impossible for a real vehicle's structure to maintain its integrity and rigidity during such maneuvers.

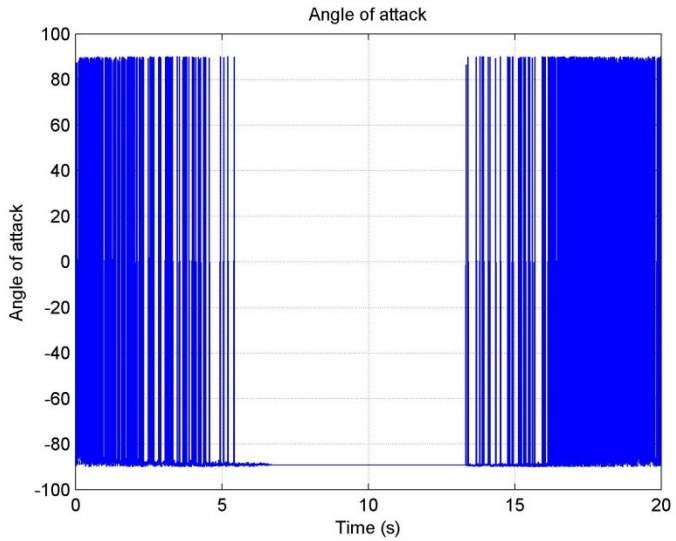

Figure 4.19 Angle of attack

It is worth mentioning that we tried to vary a lot of parameters in order to achieve a feasible trajectory, but after a lot of trials we ended up with unrealizable control parameters such as the ones shown above.

The failure of the random trajectory generator can be explained by recalling the fact that it is based on some random variables. Although these variables are bounded, they still have the ability to jump from a certain value to a totally different value in one step as long as it is within the predefined range. These jumps are probably the main reason for having the discontinuities in the control parameters.

We looked for other ways to generate a virtual feasible trajectory and we decided to use a readymade flight simulator instead of trying to mimic the flight model with a bunch of random variables which turned out to be a wrong way to generate a virtual trajectory. The flight simulator we used and the way we used it are explained in the next section.

4.3 Flight simulator based trajectory

Flight simulators are software based models for aircrafts. They rely mainly on mathematical models to simulate the aircraft behavior taking into account the aircraft structure, environmental variables such as wind, and control variables such as the engine's thrust.

Almost all sophisticated flight simulators are very expensive, but we found an open source flight simulator called *FlightGear* [37]. It supports a variety of popular platforms (Windows, Mac, Linux, etc.) and is developed by skilled volunteers from around the world. The source code is available under the GNU General Public License [38].

The goal of the *FlightGear* project is to create a sophisticated and open flight simulator framework for use in research or academic environments, pilot training, and as an industry engineering tool. It is widely considered as a realistic and challenging desktop flight simulator.

FlightGear has been used in a lot of academic studies. In [39], there is a detailed list of academic projects that used *FlightGear* in their work. The list also includes links to the projects websites in case more information is needed about a particular project. Some examples include RWTH Aachen University, university of Illinois at Urbana Champaign, Simon Fraser University, Iowa state university, and the University of Minnesota. Also in [40] *FlightGear* was used to develop an autonomous landing control scheme for a small scale unmanned helicopter.

4.3.1 Flight trajectory logging and preparation

Our aim is to fly the aircraft virtually and simultaneously log the flight data for future extraction. The parameters required to be logged are the aircraft's position (x, y, z), attitude $(roll, pitch, yaw)$, and velocity (v). This can be done easily in the *FlightGear* environment. Appendix A is a tutorial that describes the logging process in details.

Due to some limitations in the processing power of the PC used to run the simulator, the trajectory was logged with a sampling interval of $200\ ms$. Figure 4.20 shows a block diagram of the entire process of preparing the trajectory for testing the search algorithm. First the simulator is used to fly the aircraft for a certain amount of time, then the flight log which contains all the flight data is extracted from the simulator and passed to MATLAB [**14**] where all the other operations will take place.

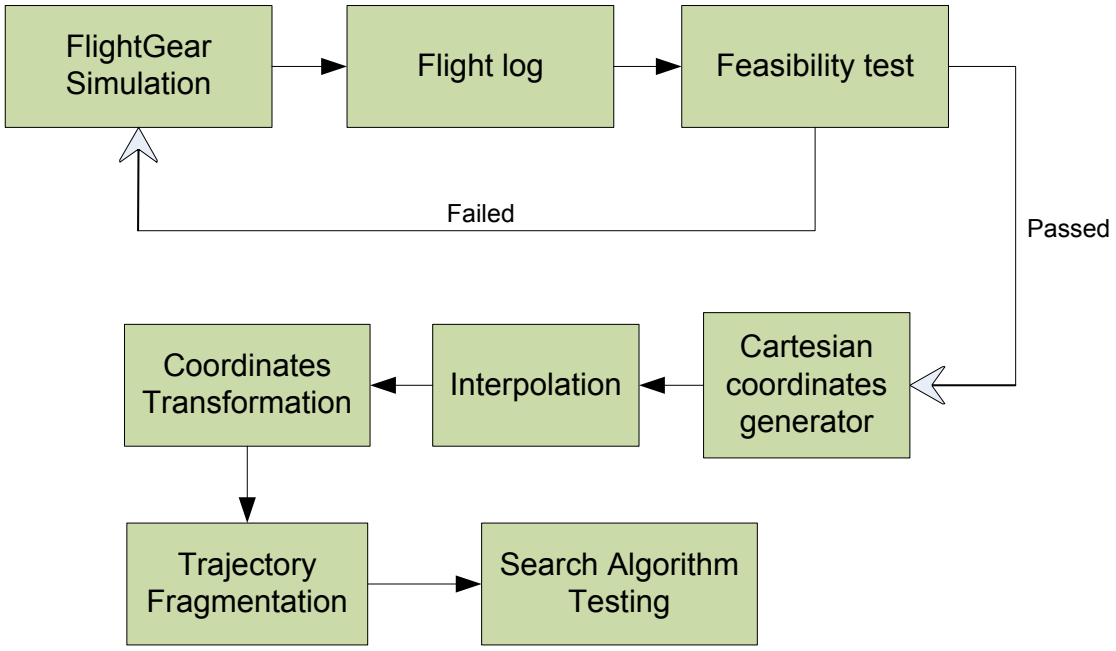

Figure 4.20 Virtual trajectory production process.

First we test the feasibility of the generated trajectory to make sure it is realizable, if not then another flight is conducted until we get a feasible trajectory. After that we adjust the units of all the data to be represented in SI units ($knots$ to m/s and $feets$ to $meters$). Then we convert the latitude and longitude readings produced by the simulator into absolute Cartesian coordinates (x, y, z), this is done using

$$d_{Lat} = Lat(i+1) - Lat(i) \tag{4.48}$$

$$d_{Lon} = Lon(i+1) - Lon(i) \tag{4.49}$$

$$x(i+1) = x(i) + R_e * d_{Lat} \tag{4.50}$$

$$y(i+1) = y(i) + R_e * d_{Lon} \tag{4.51}$$

Where d_{Lat} is the difference between two successive Latitude samples, and similarly d_{Lon} is for longitude samples, i is the samples counter, x and y are the Cartesian coordinates where $x(1) = y(1) = 0$, and R_e is the average radius of the earth in meters ($R_e = 6378137\ m$).

Then the interpolation block performs an interpolation process to reduce the intervals between samples from 200 to $5\ ms$. The reason why this reduction is necessary is explained in details in the following chapter.

After that the coordinate transformation block transforms the coordinate system into another one where the UAV is centered at its origin, this transformation simplifies the algorithm testing and helps in generating the test results quickly and easily. A detailed explanation of this transformation is provided in Appendix B.

Finally the Fragmentation block takes over to divide the 25 minutes long trajectory into a lot of small duration trajectories whose durations are 3 seconds each and finally saving

these small trajectories each in separate file. Worth mentioning that a 25 minutes flight trajectory will result in generating 500 files where $\frac{500*3}{60} = 25$.

4.3.2 The Generated trajectory

The generated trajectory is presented in this section by looking into a lot of its parameters. In Figure 4.21 the trajectory is shown in a 3D frame.

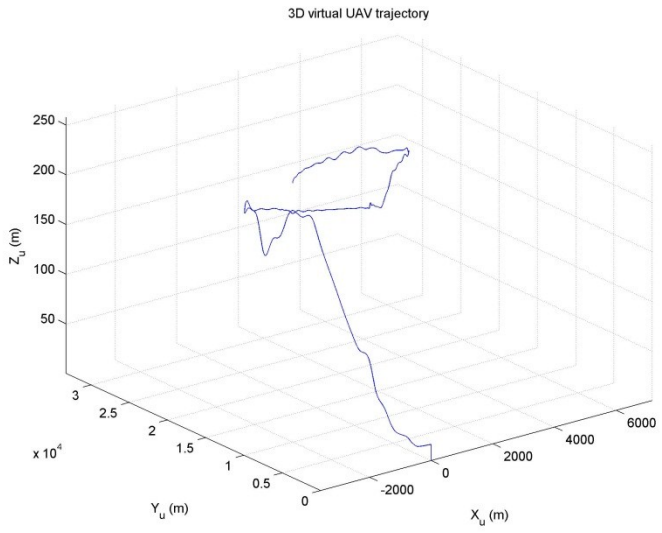

Figure 4.21 3D virtual UAV trajectory using *FlightGear*.

In order to give the reader a sense of how smooth or maneuvering the trajectory is we are showing in Figure 4.22 a top view of the trajectory shown in Figure 4.21.

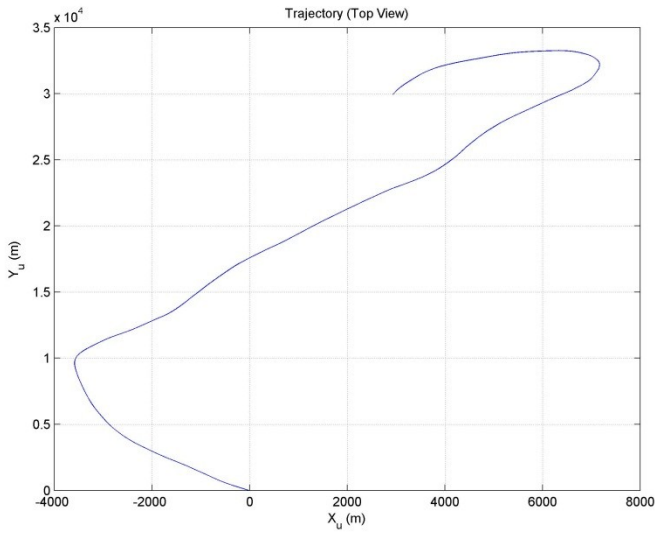

Figure 4.22 Generated virtual trajectory (Top view).

Other recorded parameters are shown as well to provide a full picture of the generated trajectory. These parameters are the altitude, velocity, and the three attitude angles $(roll, pitch, yaw)$.

In Figure 4.23 the altitude of the aircraft throughout the recorded flight time is shown. We can easily notice how hard it is for an untrained pilot to keep the altitude constant even when he is just flying a simulated aircraft. That explains why the altitude is varying between $100\ m$ and $250\ m$.

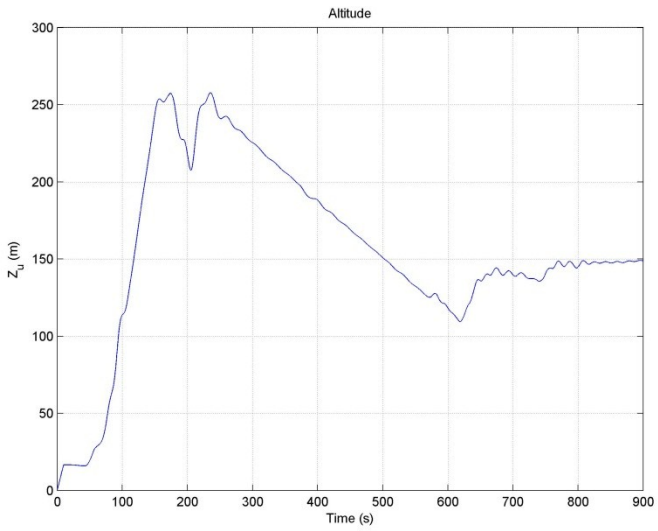

Figure 4.23 Altitude of the generated trajectory, logged by *FlightGear*.

In Figure 4.24 the velocity of the aircraft throughout the recorded flight time is shown. The values recorded by the flight simulator are compared with the values calculated from the dynamic model used to test the feasibility of the trajectory. The values recorded by the flight simulator are referred to as the measured air speed and the values calculated from the dynamic model are referred to as the calculated velocity. Notice that both curves follow the same pattern, but there is a variable shift between them. We can explain this shift by recalling the fact that the *FlightGear* simulator takes into account some environmental variables that are not considered in the dynamic model such as wind.

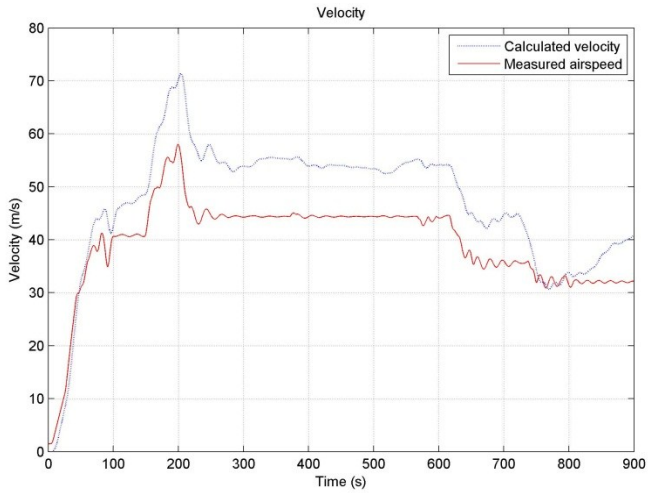

Figure 4.24 Velocity of the UAV logged by *FlightGear* and calculated by the dynamic model.

In Figure 4.25 the rolling angle (banking angle) is shown. Similar to the velocity figure we have compared the values recorded by the flight simulator (measured roll) with the values calculated from the dynamic model (calculated roll). There is a very close match.

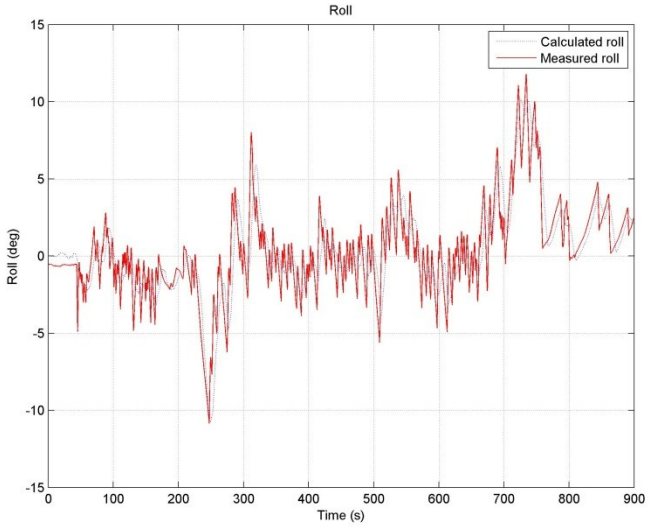

Figure 4.25 Rolling angle logged by *FlightGear* and calculated by the dynamic model.

In Figure 4.26 the pitch angle is shown. Notice how the pitching angle fluctuates along the flight time. This is the reason why we ended up with a fluctuating altitude as well. Off course a well-trained pilot can overcome this problem easily.

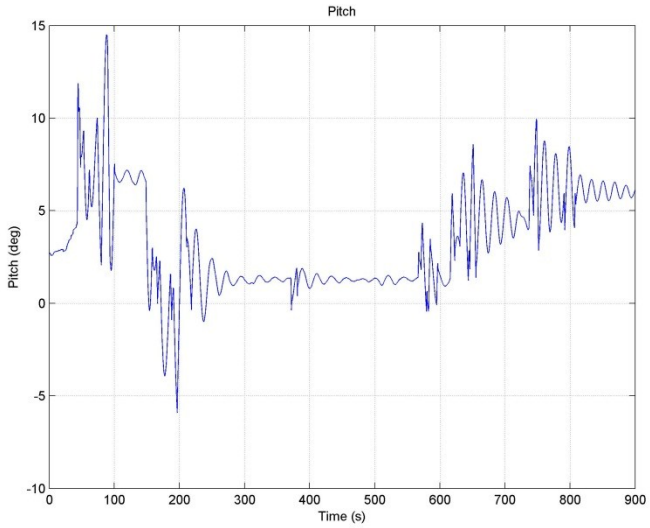

Figure 4.26 Pitch angle logged by *FlightGear*.

In Figure 4.27 the yaw angle is shown. This is considered as the heading of the aircraft where by noticing the changes in the yaw angle one can easily link between the changes in the yaw and the turns made by the aircraft throughout the flight.

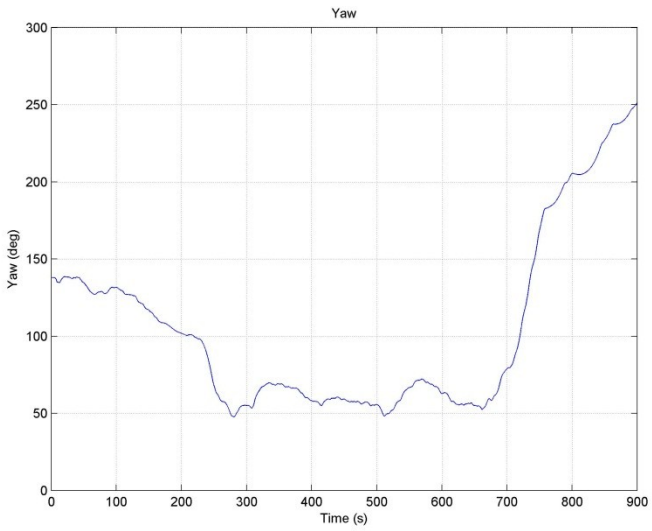

Figure 4.27 Yaw angle (aircraft heading) logged by *FlightGear*.

4.3.3 Feasibility test

In this section we will repeat the test we made before for the randomly generated trajectory except that this time we will do it for the flight simulator based trajectory. We will show the difference between a feasible realizable trajectory and an unfeasible unrealizable one.

We will start by observing the acceleration pattern throughout the flight time where it is shown in Figure 4.28. We would like to draw the reader's attention to the fact that the pattern is relatively smooth and contains no discontinuities or unbounded overshoots as it was the case in the previous acceleration pattern showed in Figure 4.18. Notice that the curve covers 900 seconds (15 minutes) of flight time.

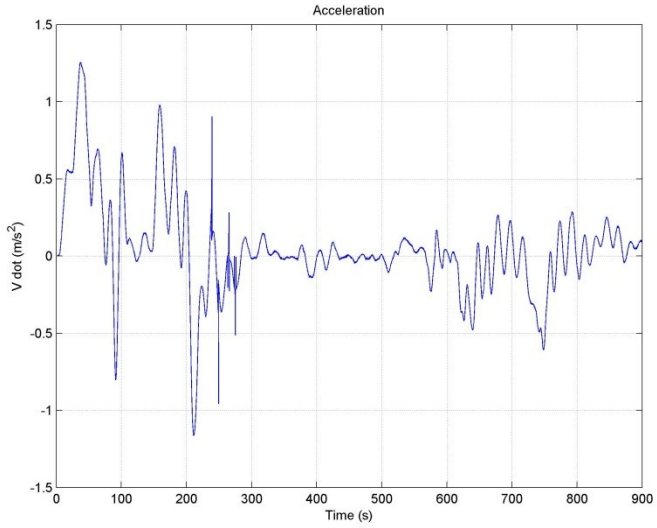

Figure 4.28 Acceleration pattern of the trajectory generated by *FlightGear*.

In Figure 4.29 the thrust pattern of the aircraft throughout the flight time is shown. One can easily observe that the pattern is totally unlike what was showed in Figure 4.17 where this time the thrust is continuous, bounded and relatively smooth.

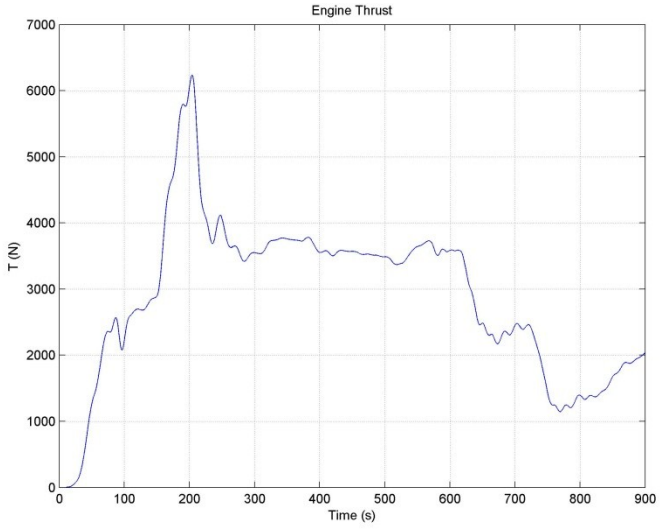

Figure 4.29 Engine's thrust.

These observations can lead to a strong statement of considering the trajectory generated by the flight simulator as a feasible and a realizable trajectory. Thus it can be validated

for testing our search algorithm where the performance evaluation results will now depend only on the algorithm performance and nothing else. This performance evaluation process is explained in details in the next chapter.

4.4 Conclusions

In this chapter we presented two methods of generating a virtual UAV flight trajectory. The first method was the random trajectory generator. The trajectories created by this generator didn't pass the feasibility test because it produced sharp discontinuities in the control variables such as the engine thrust due to the randomness of the variable distribution that gives independent outputs each run. The second method was using a flight simulator (*FlightGear*) to fly the aircraft manually and log the flight data for further analysis and processing. The trajectories produced by this method passed the feasibility test where it produced continuous realistic thrust pattern which can be produced by a real engine. Thus trajectories generated by the flight simulator proved their validity to be used in testing the search and tracking algorithms.

Chapter 5: Search Algorithm

5.1 Introduction

At any given time instant, the UAV will have a certain heading for its antenna beam. If the beam heading is misaligned with the direction of the maximum RSS, then this misalignment will cause degradation in the wireless signal strength. On the other hand if the beam heading is aligned with the direction of the maximum RSS, then the signal strength will improve and the communication link will be enhanced significantly.

The function of the search algorithm is to adjust the beam heading angles (θ_b, φ_b) such that the antenna beam will be steered to the direction of the maximum RSS.

Notice that the antenna is installed on a flying UAV. This will lead to a highly dynamic communication channel and a highly dynamic RSS distribution in space. Thus the algorithm is required to be fast enough to cope with this dynamic behavior. Also it is required to be simple enough to be executed by a small hardware platform that can be installed on a low payload UAV.

The search algorithm will conduct a series of RSS measurements and according to these measurements it will produce an estimate for the best beam heading angles (θ_b, φ_b) which produces the maximum achievable RSS.

For this algorithm we must define our search space. The space in which we are searching for the maximum RSS is a spherical plane. Thus in a spherical coordinate system our plane of interest is defined by

$$0° < \theta_b < 180° \tag{5.1}$$

$$0° < \phi_b < 360° \tag{5.2}$$

$$r = constant \tag{5.3}$$

5.2 Previous work

Localization techniques were summarized and presented in [41], and [42]. Any localization technique is based on a certain signal parameter being measured. The most common of these signal parameters are the angle of arrival (AOA), time of arrival (TOA), time difference of arrival (TDOA), and the received signal strength (RSS).

Most angle of arrival (AOA) estimation techniques rely on extracting the received signal from the output of several signal receivers whose antennas are separated by a fraction or few wavelengths, and deployed in a special arrangement. Any signal arriving at any incident angle other than 90˚ will be detected by different signal sensors at slightly different times. Therefore the signal received will contain time and phase delayed versions of the same signal. By processing these different versions of the same signal we can extract the corresponding angle of arrival (AOA) of the signal. Thus by doing this at a number of different locations we can combine AOA measurements in a Multilateration algorithm to eventually localize the signal source.

In [43], an outdoor localization method was presented, it was successful in locating the node of interest even in non LOS (line of sight) situations, the localization was based on the AOA and utilized multiple reference stations.

Time of arrival (TOA) is directly related to the propagation time which is a function of the channel medium and the signal frequency. If the medium is known and a LOS situation is proposed, then the distance between the transmitter and the receiver can be directly related to the propagation time. By taking several TOA measurements at different locations, a location estimate for the transmitter can be produced. This TOA

measurement requires a very good synchronization between the transmitter and all the receivers.

Time difference of arrival (TDOA) is similar to TOA but with eliminating the need for synchronization between the transmitter and the receivers. TDOA require synchronization between the receivers only.

The RSS has a measurement variance that increases with the range [44]. But all of the above mentioned signal parameters require more than one receiver to obtain the measurement except for the RSS. Since we are constrained by a small payload of our vehicle and the availability of only one transmitter and one receiver, therefore the only parameter available for measurement in our case is the RSS. Hence we will only look into localization methods based on the RSS measurements.

Localization based on RSS is greatly dependent on the propagation model used. Usually it is required to validate the propagation model by tuning the path loss exponent (distance-power gradient). This is done by characterizing the situation through a lot of RSS measurements. Thus practical implementation is a must in this case. Since we have a dynamic channel, it is not possible to characterize the link with a valid characterization which can support an accurate propagation model.

In [45], different RSS based localization methods were compared in terms of the calibration effort required by each method. In [46], the localization was based on RSS, yet it was utilizing multiple reference stations like [43]. In [47] the characteristics of an RSS signal was studied and characterized, where it was obvious that the RSS value will change dramatically by changing the environmental circumstances.

In [48], a successful attempt was made to estimate the path loss exponent jointly along with the location estimation process however, this estimation procedure is a computationally intensive task., hence it will not be suitable for our application.

In this work we will propose two techniques utilizing the RSS readings in localizing the maximum signal strength by steering the antenna beam to different directions. These are the elliptical Peeking (EP) and the differential evolution (DE) algorithms. The EP algorithm is proposed as a contribution in this work and the DE algorithm is used to benchmark the performance of the EP algorithm. Both techniques will require a simulation environment that will allow us to calculate the RSS at a certain instant within the virtual trajectory generated in chapter 4. Section 5.3 presents the procedures carried out in the simulation to calculate the RSS.

5.3 Simulating the Received Signal Strength Indicator (RSSI)

Simply this is the basis of a *MATLAB* [28] function used in the simulation to produce RSSI measurements. In order to produce these readings virtually we must consider a full communication link analysis. Figure 5.1 illustrates the inputs and the outputs of this function. The function takes two inputs. First the position of the ground transmitter (X_t, Y_t, Z_t) relative to the UAV by considering a UAV centered frame of reference. Then the antenna beam steering angles (beam heading) (θ_b, φ_b), and it produces the received power P_r in dB as an output.

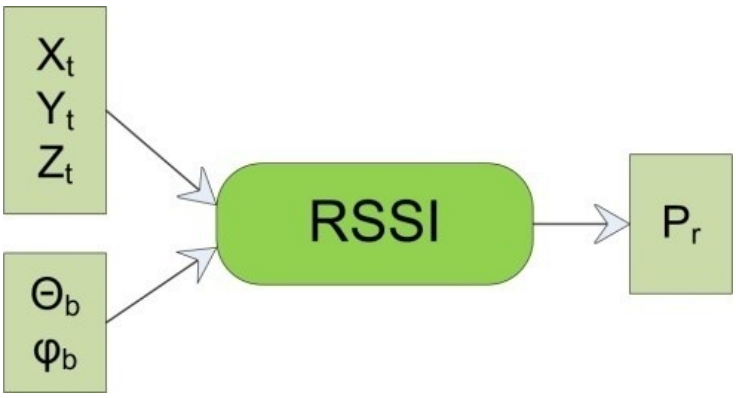

Figure 5.1 RSSI readings generating function with its inputs and outputs.

The communication link in this case consists of a ground transmitter, a channel, and a UAV embedded receiver [22]. Our link analysis is based on the path loss model in (5.4), assuming zero losses in the feeding cables of the transmitter and the receiver.

$$P_r = P_t + G_t - P_l + G_r \tag{5.4}$$

Where P_r is the received power, P_t is the transmitted power, G_t is the gain of the transmitting antenna (ground station), P_l is the power losses in the communication channel, and G_r is the gain of the receiving antenna. All of these quantities are in dB.

The transmitting antenna is assumed to have an isotropic radiation pattern with zero dB gain ($G_t = 0$). The transmitted power is $10\,dBm$ according to the transmitter's specifications [22]. The power losses can be calculated using

$$P_l = 20 * \log\frac{4\pi R}{\lambda} \tag{5.5}$$

Where λ is the wave length and R is the line of sight distance between the transmitter and the receiver, in other words the ground station and the UAV.

Since our operating frequency is 2.45 GHz, therefore the wavelength λ is $122.4\,mm$ according to equation (5.6).

$$\lambda = \frac{c}{f} \tag{5.6}$$

Notice that f is the frequency and c is the speed of light ($299{,}792{,}458\ m/s$).

The line of sight (LOS) distance R can be calculated using

$$R = \sqrt{X_t^2 + Y_t^2 + Z_t^2} \tag{5.7}$$

The gain G_r of the receiving antenna array embedded inside the wing structure of the UAV is calculated through two steps. The first step is to evaluate the radiation pattern based on the steering angles of the main lobe of the beam (θ_b, φ_b), thus we will have a matrix of gain values corresponding to a full scan in the $(\theta - \varphi)$ space, the matrix will take this form

$$G_r(\theta, \phi) = \begin{bmatrix} G_r(1°, 1°) & \cdots & G_r(1°, 360°) \\ \vdots & \ddots & \vdots \\ G_r(180°, 1°) & \cdots & G_r(180°, 360°) \end{bmatrix}_{180\times360} \tag{5.8}$$

The second step is to choose out of all these gain values (G_r) the one whose (θ, φ) corresponds to the true transmitter heading angles (θ_t, φ_t). This choice can be performed by first calculating (θ_t, φ_t), then choosing the $G_r(\theta, \phi)$ value whose (θ, φ) is the closest to (θ_t, φ_t). Since our radiation pattern is generated with a resolution of 1°, therefore any approximations will not have any significant effect on the value of the gain.

Since we already have the ground station coordinates in a Cartesian form (X_t, Y_t, Z_t) after applying the transformation in appendix B, the transmitter's heading angles (θ_t, φ_t) can be obtained after a coordinate transformation from Cartesian to spherical coordinates.

This spherical coordinate system is illustrated in Figure 5.2 (same as Figure 3.1) and the conversion is performed by applying equations (5.9), and (5.10). Note that both coordinate systems (Cartesian and spherical) have the same origin and the UAV is centered at this origin.

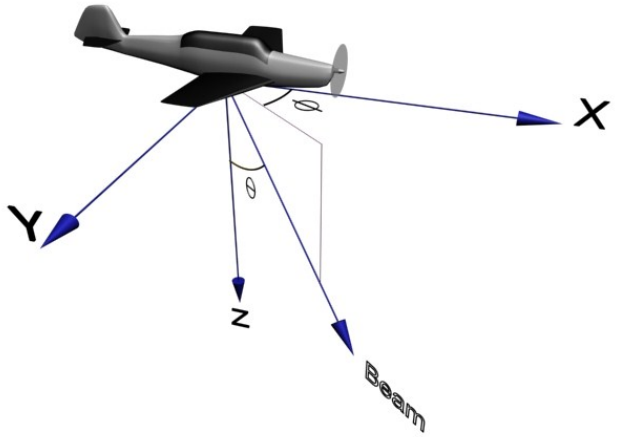

Figure 5.2 spherical UAV-centered coordinate system.

$$\theta_t = \cos^{-1}\frac{Z_t}{R} \tag{5.9}$$

$$\varphi_t = \tan^{-1}\frac{Y_t}{X_t} \tag{5.10}$$

Where R is the LOS distance given by equation (5.7).

The previous procedures will provide all the terms required in equation (5.4) which will be used to calculate the received power P_r.

In the following three sections we will explore three possible ways of finding the direction of the maximum RSS. The first one (Ranging) is well known in literature, but it does not fit to our problem. The second one (elliptical Peeking) is proposed by this work

and showed very good results. The third one (Differential evolution) is a well-known intelligent optimization algorithm that is usually not used for real time optimization problems. We applied it to our problem and it showed good results, but not as good as elliptical Peeking.

5.4 Ranging

Ranging is a popular technique of localizing targets. It is the main concept behind the GPS system. It can depend upon several signal parameters (AOA, TOA . . . etc.). We will discuss it based on the RSS.

RSS based ranging is performed by measuring the RSS at a number of different locations around a stationary transmitter, then using a propagation model to calculate the range between the transmitter and the receiver for each measurement. A simple example for the propagation model is stated in equations (5.4) and (5.5). The transmitter's relative position can then be deduced by multilateration. This technique is well explained in [41]. It is also well known in literature where in [49] the localization accuracy was improved by using a lognormal shadowing propagation model and a weighted least squares approximation technique.

In our case we can utilize the mobility of the UAV to take different RSS measurements from different locations. Figure 5.3 shows a description of the situation in 2D.

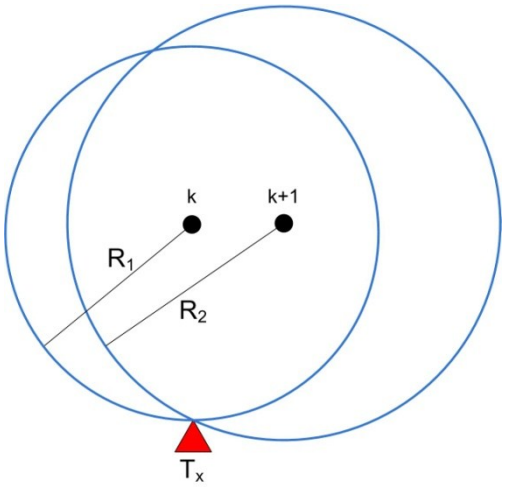

Figure 5.3 2D RSS ranging based source signal localization situation

In Figure 5.3 the black dots resembles UAV trajectory samples, R_1 and R_2 are the ranges estimated using the propagation model. We can generalize this method for 3D situations. We will then need 4 RSS reading. Though we will usually employ more than 4 readings because the RSS measurements are usually noisy and their variance increases directly with the increase in range. Therefore we can smooth our estimate by employing more measurements using a statistical Multilateration algorithm [41].

The above Localization technique requires an isotropic receiving antenna and an isotropic transmitting antenna. An isotropic transmitting antenna is already in our assumptions. For the receiving antenna it is not possible to generate an isotropic radiation pattern using the antenna array designed earlier. Also it requires a realistic propagation model which is usually a complicated task because it requires a lot of measurements for tuning the model parameters. Also even after tuning the propagation model parameters, it will be outdated if the UAV moved from an open wide environment to an urban environment which is considerably possible in some missions. Thus ranging will not be our method of choice.

5.5 Elliptical Peeking Algorithm (EP)

5.5.1 Description

This is an RSS based Beam steering algorithm that is considered one of the contributions of this work. The strongest signal source exists somewhere in the $(\theta - \varphi)$ space. At a certain time instant within the trajectory, we pause the time and measure the RSS distribution in the $(\theta - \varphi)$ space. By scanning all the angles for RSS values we produced the contour plot shown in Figure 5.4. This plot is generated by conducting an exhaustive search at a given time instant which requires a lot of RSS measurements and since one measurement requires $5\,ms$, therefore producing this scan consumes a lot of time. It would be totally inappropriate to consider exhaustive search in our situation, because by the time one complete scan is produced, the vehicle would have moved to another location and will have different attitude .The produced scan would be inapplicable to the new location and attitude.

Let us imagine that by directing the beam towards a certain point in the $(\theta - \varphi)$ space and measuring the RSS at this point, we are Peeking at the RSS of this point. Thus by collecting a number of RSS measurements from a number of locations in the $(\theta - \varphi)$ space, we are Peeking at the RSS of a number of points in the space. The points at which we are Peeking are divided into groups, where each group of points forms an elliptical shape as shown in Figure 5.7, hence the name elliptical Peeking (EP). Notice that the $(\theta - \varphi)$ space is originally a curved space represented by the surface of a sphere, but here we are showing it as a rectangular space for simplicity.

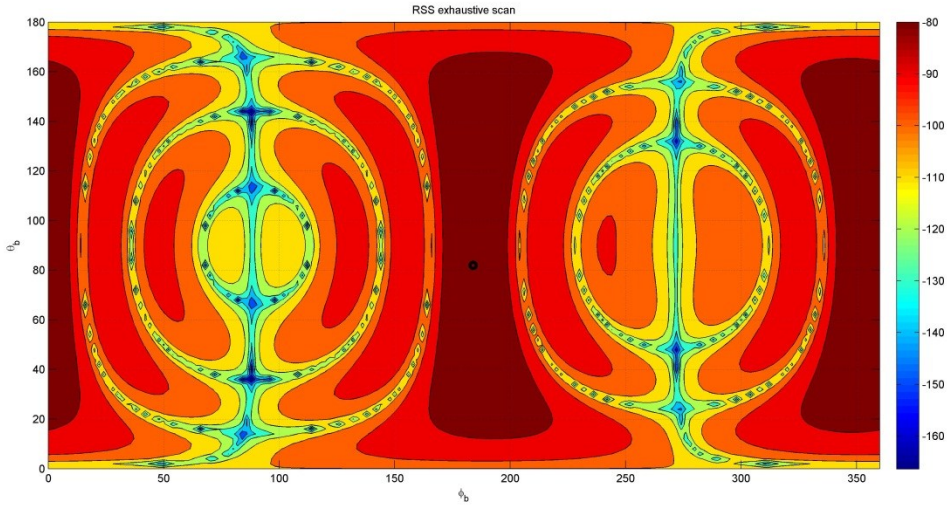

Figure 5.4 Contour plot for the measured RSS values in the θ-φ space at a certain time instant. The black circle denotes the maximum RSS value measured. The color bar represents the RSS values in dBs

The flow chart of the algorithm is shown in Figure 5.5. After the algorithm initializes, it constructs the ellipse points at which the RSS readings will be collected, and then it collects them all and saves them in the memory. After that it compares between all of these RSS values and picks the greatest among them and sets its corresponding beam steering angles (θ_b, ϕ_b) as the center of the next ellipse. But before it goes into the loop

again, it checks for the termination conditions in order to decide whether to complete the search process or to stop and produce its (θ_b, ϕ_b) estimate to the tracking routine. The tracking routine is supposed to keep track of the values of (θ_b, ϕ_b) which gives the maximum possible RSS by conducting a very fey number of iteration from the search algorithm every once and a while. The time slots at which the tracking routine will conduct its mini-search will depend upon the communication protocol followed by the UAV transmitter. This is because the tracking routine will utilize the time periods at which the transmitter is idle in order to switch to the receiving mode and extract the necessary RSS readings required for the search, keeping in mind that the ground station is sending a continuous tracking signal. This signal will be available all the time to allow the UAV to extract RSS reading at any instant. The tracking routine is discussed in Chapter 6.

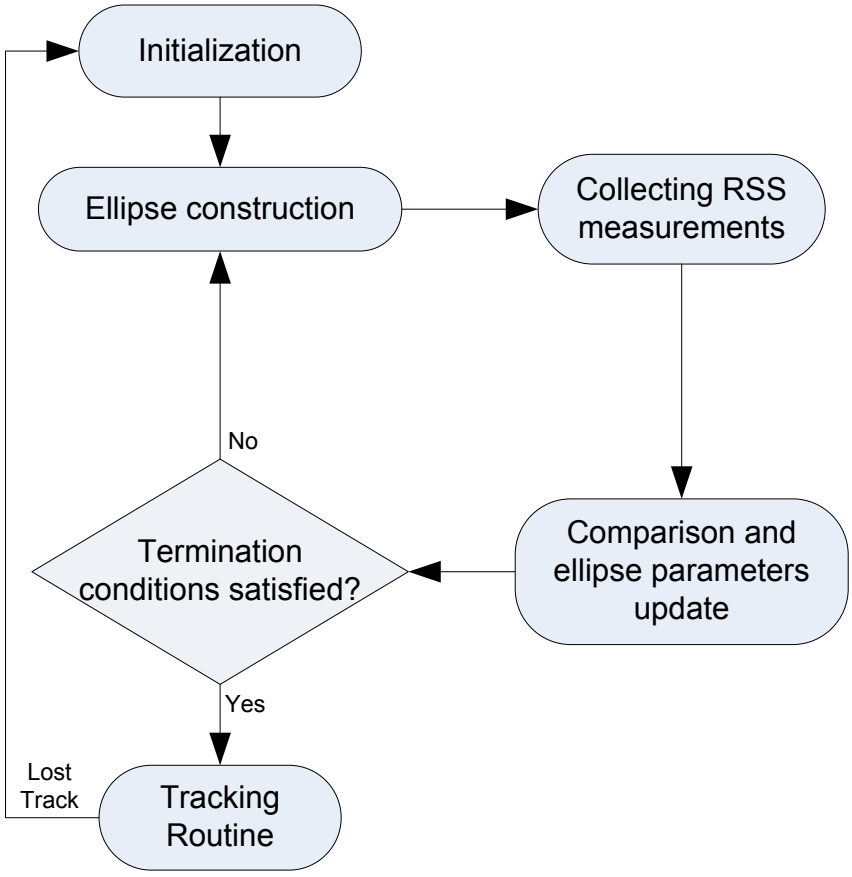

Figure 5.5 Flow chart of the elliptical Peeking (EP) algorithm

- **Initialization**

The initialization starts the EP algorithm by executing three consecutive procedures. These procedures are shown in Figure 5.6. The first procedure sets the values of the algorithm parameters (f_n, f_r, td, tl). These four parameters define how the algorithm will be executed. They can be tuned for better performance in different situations. The values we used were achieved by trial and error and they are as follows:

$$f_n = 0.5 \quad (5.11)$$

$$f_r = 0.8 \quad (5.12)$$

$$t_d = 0.4\ dB \quad (5.13)$$

$$t_l = 2\ s \quad (5.14)$$

Where the factor f_n is the reduction factor of n, where n is the number of ellipse points. This is the factor by which the number of ellipse points will be reduced every loop turn. f_r Is the r_ϕ reduction factor which is the factor by which the ellipse radius r_ϕ will be reduced every iteration. Note that the ellipse has two radii ($r_\phi \, and \, r_\theta$) as shown in Figure 5.8. t_d Is the signal strength (SS) termination difference where this is the minimum improvement in the SS that will allow the algorithm to go for an extra iteration. In other words, if the improvement in the SS is less than t_d then the algorithm will terminate the search. t_l Represents the time limit at which the algorithm will terminate its search regardless of anything else and produce its last estimate as its best estimate. This condition is there to protect the algorithm from going into an infinite loop.

The second procedure in the initialization process is to define the search space. This is an easy step where all we have to do is to set the values for θ_{max}, θ_{min}, ϕ_{max}, and ϕ_{min}. The limits of the search space are defined by inequalities (5.15) and (5.16).

$$0° \leq \theta \leq 180° \tag{5.15}$$

$$0° \leq \phi \leq 360° \tag{5.16}$$

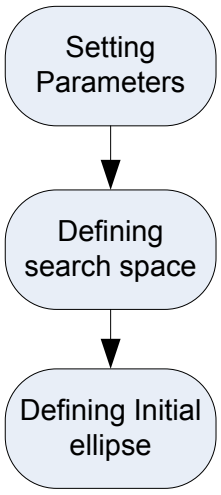

Figure 5.6 Initialization procedures for the EP algorithm.

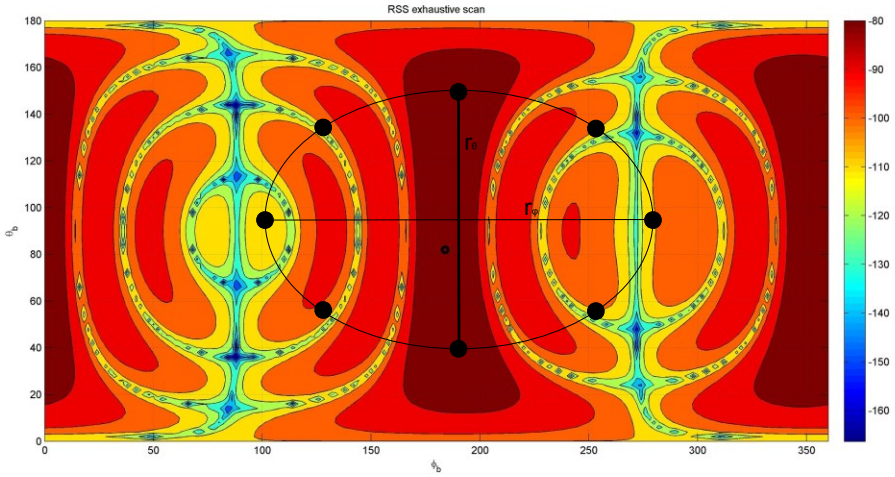

Figure 5.7 A graphical illustration of the initial iteration in the EP algorithm.

The third and final procedure in the initialization process is defining the initial ellipse. According to Figure 5.7 the ellipse can be completely defined by four parameters, the ellipse center, the major radius, the minor radius r_a, and the number of points n in the ellipse edge. These parameters are initially defined according to the minimum $HPBW$ of the used antenna array. This is done to make sure that the first ellipse will have at least one point inside the footprint of the antenna's main lobe. Equation (5.17) defines the initial value of r_ϕ. Equation (5.18) can be used consequently to calculate the initial value of r_θ, where the ratio stated in this equation must hold through all the iterations. Thus we also use equation (5.18) to calculate r_θ in the middle of the algorithm execution whenever r_ϕ changes as a result of being multiplied by the reduction factor f_r. Equation (5.19) is used to calculate the initial value of n.

$$r_{\phi_{initial}} = \frac{1}{2}\left[\phi_{span} - \frac{HPBW_{min}}{2}\right] \tag{5.17}$$

$$\frac{r_\theta}{r_\phi} = \frac{\theta_{span}}{\phi_{span}} \tag{5.18}$$

$$n_{initial} = \frac{r_{\phi_{initial}}}{\frac{1}{2}HPBW_{min}} \tag{5.19}$$

- **Ellipse Construction**

Ellipse construction is the process of adjusting the number of ellipse points n and generating their corresponding beam steering angles (θ_b, ϕ_b). In Figure 5.8 the construction procedures are shown. The first step is to make sure that n is divisible by 4 in order to construct a symmetric ellipse at the end. Thus n in this step is approximated to the nearest greater number divisible by 4. The second step protects n from being reduced below a value of 8. The last step is the process of calculating the beam steering angles (θ_b, ϕ_b) for all the ellipse points in preparation for measuring the RSS at each one of these angles.

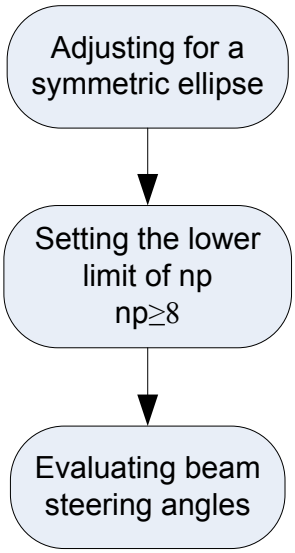

Figure 5.8 Ellipse construction procedures

- **Comparison and Update**

The comparison and update block is unfolded in the flow chart shown in Figure 5.9. After collecting a set of RSS readings we pick the highest value. Around the picked value we collect another group of RSS readings and also pick the highest. Then we can repeat these steps to fine tune the position estimate to an acceptable accuracy.

We will try to close on the maximum RSS keeping in mind that there are lots of local maxima in the search space. Our proposed algorithm is based on an iterative process that starts at the global level and is expected to evolve to close on the global maximum RSS, where further iterations will allow the algorithm to fine tune its estimation and improve accuracy, this will cost more time which is a critical factor here due to the dynamic situation at hand. A compromise will be made to reach an acceptable accuracy within a reasonably short period of time.

Updating n and r_ϕ is performed according to equations (5.20) and (5.21) where k is the algorithm iteration counter. Updating r_θ is performed the same way it was initialized by equation (5.18).

$$n(k+1) = n(k) * f_n \tag{5.20}$$

$$r_\phi(k+1) = r_\phi(k) * f_r \tag{5.21}$$

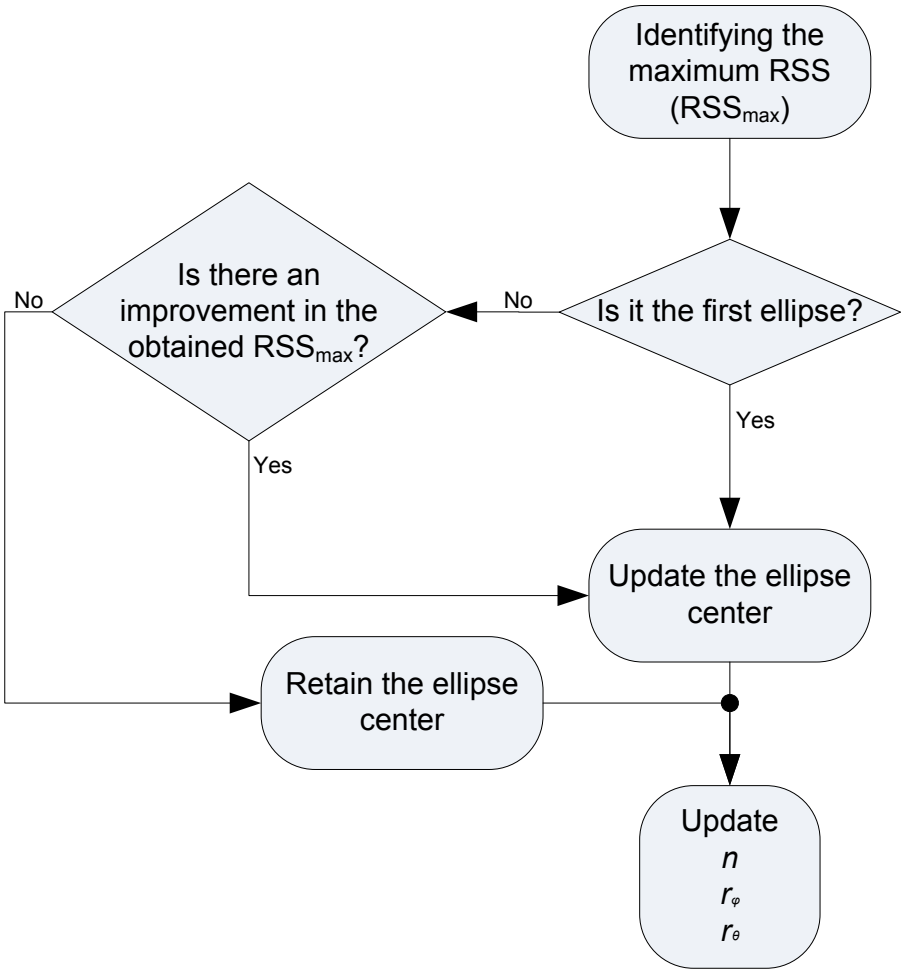

Figure 5.9 Comparison and update process block diagram.

5.5.2 Simulation Results

In this section we will present the simulation results of the EP algorithm, and we will evaluate its performance by simulating a large number of flight trajectories (500) for two times, one time using a smooth nearly straight flight trajectory and another time for a turbulent flight trajectory with a lot of tight maneuvers. This is done to assess the algorithm performance in all flying circumstances.

5.5.2.1 Smooth trajectory Results

In Figure 5.10 a top view of a smooth trajectory is shown. One can observe that the trajectory is relatively smooth with no tight maneuvers or turns.

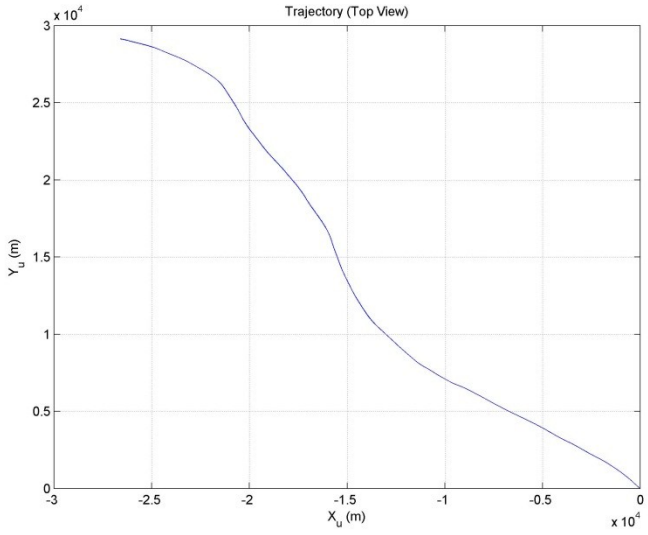

Figure 5.10 Top view of the used *FlightGear* trajectory (smooth trajectory).

Figure 5.11 shows the simulation results for the 500 mini-trajectories created from the big trajectory shown in Figure 5.10. For each one of the 500 runs we have two RSS values, one represents the RSS achieved by the algorithm and the other represents the maximum achievable RSS produced by a perfectly aligned beam. It is obvious that in almost all the points the two readings coincide together indicating a successful convergence to the maximum achievable RSS. Further error analysis is performed in order to quantify the errors and have a deeper sense of the performance.

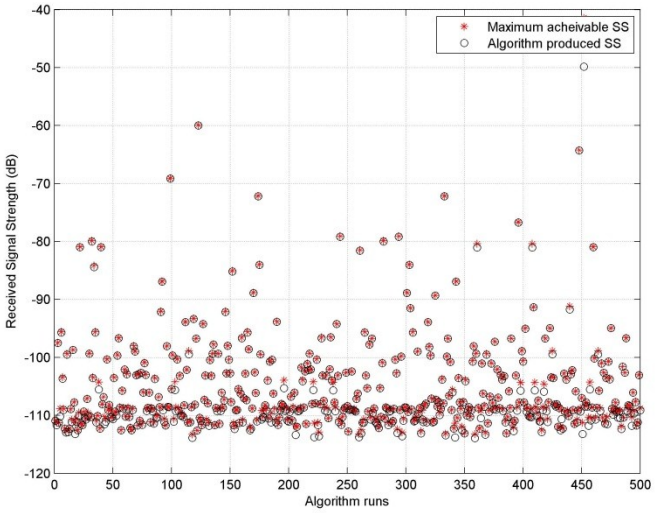

Figure 5.11 The maximum achievable RSS versus the RSS obtained by the EP algorithm (smooth trajectory).

Figure 5.12 shows the RSS values obtained by the EP algorithm and the maximum achievable value in each one of the 500 runs. The algorithm error in each run is the difference between the red stars and the black circles in Figure 5.11. Notice that the average error value is $-0.24\ dB$ and the maximum error doesn't exceed $-3\ dBs$ with only one anomaly at $-8.4\ dB$. In Figure 5.13 a histogram of the RSS error values are shown to provide the reader with the sense of how the values are distributed among the signal strength error scale. One can easily observe that more than 94% of the runs have error values within $-0.84\ dB$.

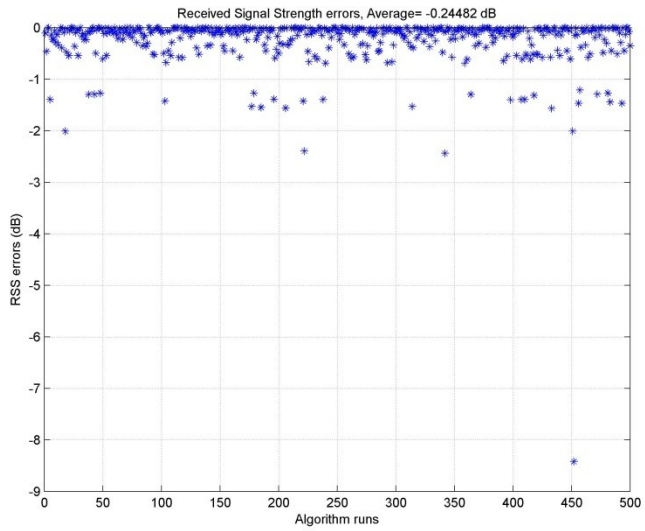

Figure 5.12 Errors in the RSS obtained by the EP algorithm through the 500 trajectories (smooth trajectory).

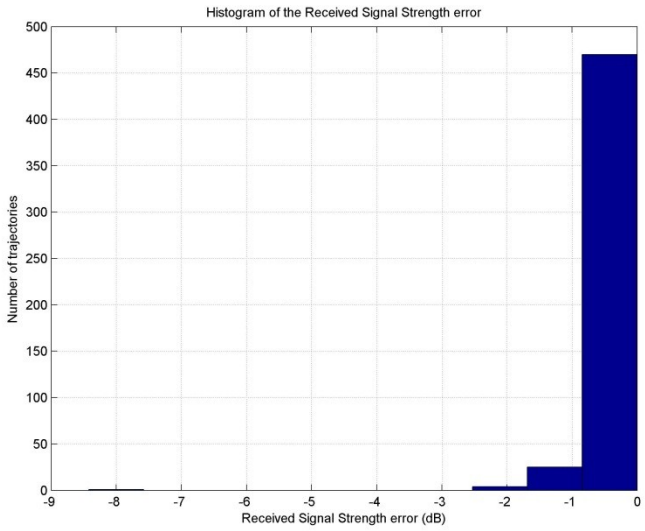

Figure 5.13 RSS errors histogram (smooth trajectory).

Figure 5.14 shows the Time of convergence (TOC) for the 500 runs. The time of convergence is the time period consumed by the algorithm before producing its output. The average TOC is $431\ ms$ and there are 3 runs out of 500 where the algorithm was terminated by the time limit condition instead of the RSS improvement condition defined by t_d from equation (5.13). In Figure 5.15 the histogram of the TOC is shown to give the

reader a sense of how the TOC values are distributed on the time scale, where we notice that 99.2% of the 500 trials converged within the $620\,ms$ limit which is a little more than the average TOC value.

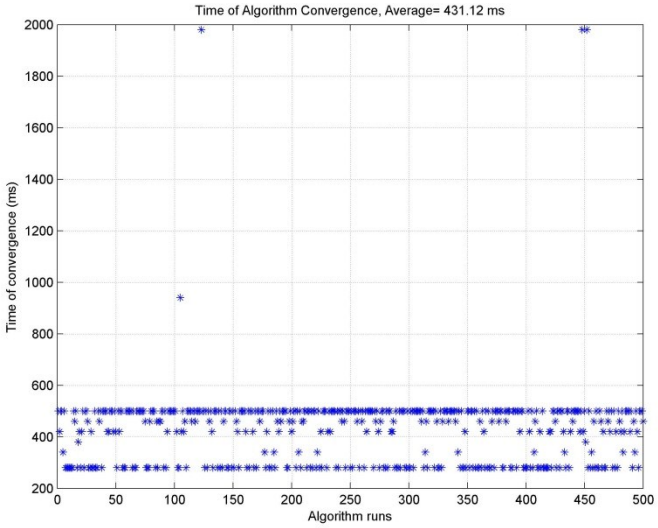

Figure 5.14 Time of convergence (smooth trajectory).

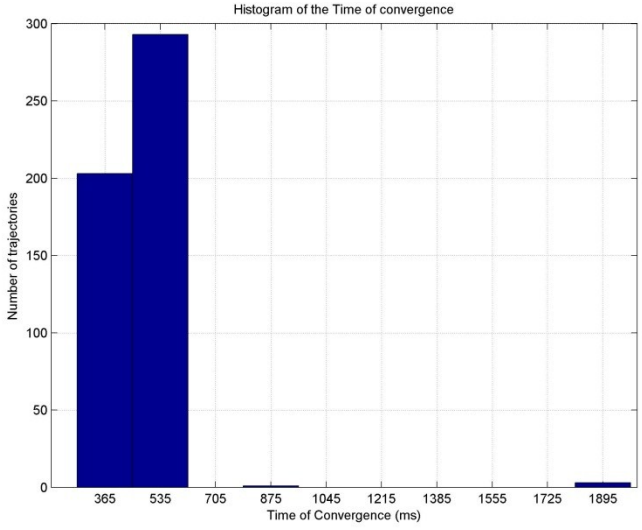

Figure 5.15 Histogram of TOC (smooth trajectory).

In Figure 5.16 the errors in estimating the beam steering angle θ_b are shown. Notice that the errors have an average value of 3.5°. This is relatively low compared with the average

error value of ϕ_b shown in Figure 5.17 which is $72.5°$. This can be easily explained by referring to the antenna array chapter and recalling the fact that $HPBW_\theta$ is usually smaller than $HPBW_\phi$. This will result in higher localization accuracy in the θ axis and a less localization accuracy in the ϕ axis. This also assures the fact that this algorithm equipped with the designed antenna cannot be used for localization or navigation unless we use a different antenna array. Thus it can be used only for communication link enhancements because it has very good abilities of maximizing the RSS.

By carefully observing the simulation results, we deduced that errors in estimating the steering angles don't degrade the RSS because of two reasons. The first reason is that the antenna beam is split into two beams, and sometimes the EP algorithm maximizes the RSS by aligning the secondary beam to the incoming signal instead of the primary beam. Since the two beams don't have a considerable difference in the gain value, therefore the RSS will not be affected much if the secondary beam was pointing at the signal source instead of the primary beam. This is why the errors in ϕ_b exists around only two values ($0° \; and \; 160°$) where the ones around $0°$ are obtained by the primary beam, while those around $160°$ are obtained by the secondary beam. The second reason is that the beam has a wide footprint within which the gain doesn't have considerable changes. So as long as the maximum RSS point falls inside the footprint the RSS value will be acceptable and the algorithm will terminate itself.

If we were to use an antenna array with low HPBW then any small error in the steering angles will have a great effect on the RSS value, because in this case the footprint will be

small. This can be utilized for localization or navigation reasons, but one has to keep in mind that the TOC in this case might increase.

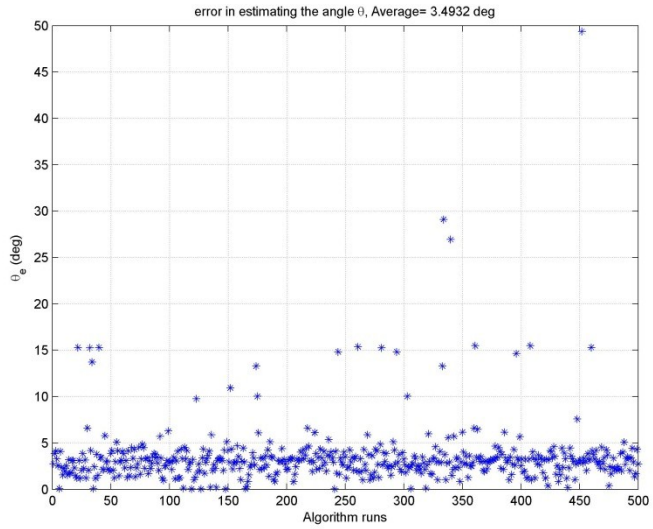

Figure 5.16 Errors in estimating θ_b (smooth trajectory).

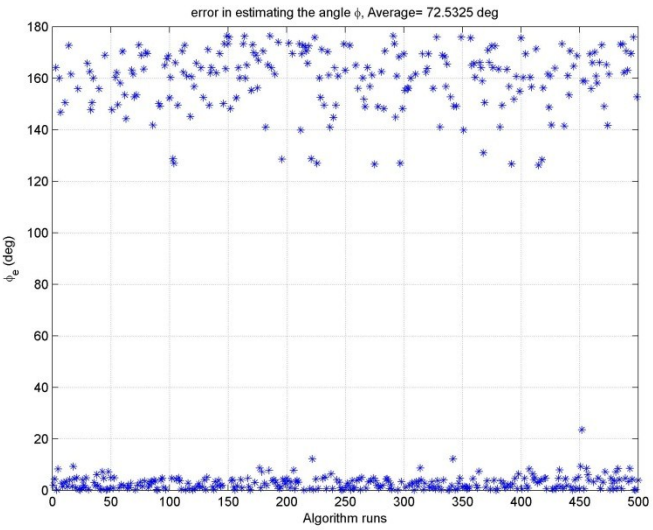

Figure 5.17 Errors in estimating ϕ_b (smooth trajectory).

5.5.2.2 Turbulent trajectory Results

The same simulation we conducted on the smooth trajectory was performed on a turbulent one. A different flight trajectory with a lot of tight maneuvers was used. In Figure 5.18 a top view of the trajectory is shown.

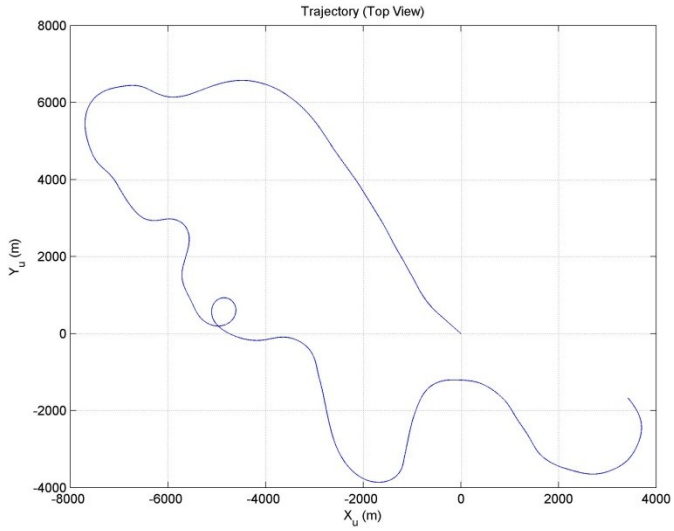

Figure 5.18 Top view of the used *FlightGear* trajectory (Turbulent trajectory).

The other results are shown in the same order they were presented in the smooth trajectory section. One can easily observe that the algorithm has good performance in a turbulent trajectory just like a smooth one.

Figure 5.19 shows the simulation results for the 500 mini-trajectories created from the big trajectory shown in Figure 5.18. For each one of the 500 runs we have two RSS values, one represents the RSS achieved by the algorithm and the other represents the maximum achievable RSS produced by a perfect beam steering. It is obvious that in almost all the

points the two readings coincide together indicating a successful localization of the maximum achievable RSS.

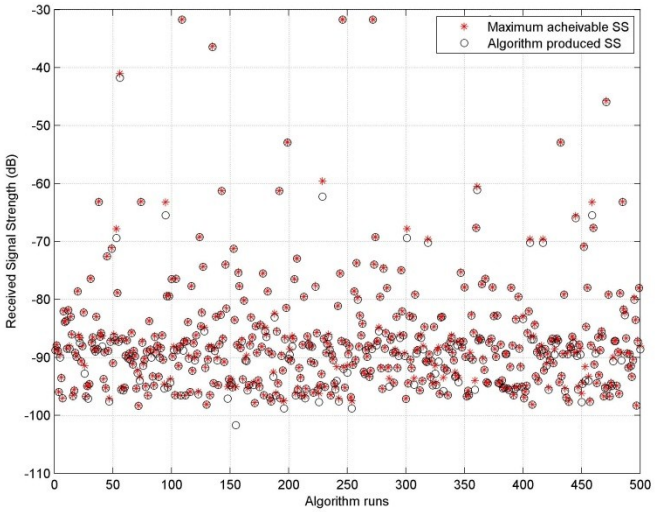

Figure 5.19 The maximum achievable RSS versus the RSS obtained by the EP algorithm (turbulent trajectory).

Figure 5.20 shows the errors in the RSS. This is the difference between the maximum achievable RSS and the RSS achieved by the algorithm. Notice that the average error value is $-0.27\,dB$ and the maximum error doesn't exceed $-2.5\,dBs$ with a single exception at the run 147 which produced an error of $-26.69\,dB$. This is a single anomaly out of 500 runs. Thus it doesn't have a considerable effect on our performance assessment. In Figure 5.21 a histogram of the RSS error values are showed to provide the reader with the sense of how the values are distributed among the signal strength error scale. One can easily observe that more than 95% of the runs have error value within $-1.5\,dB$.

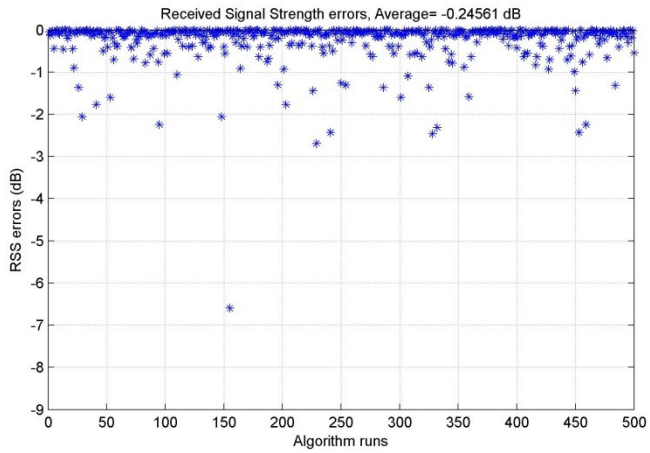

Figure 5.20 Errors in the RSS obtained by the EP algorithm through the 500 trajectories (turbulent trajectory).

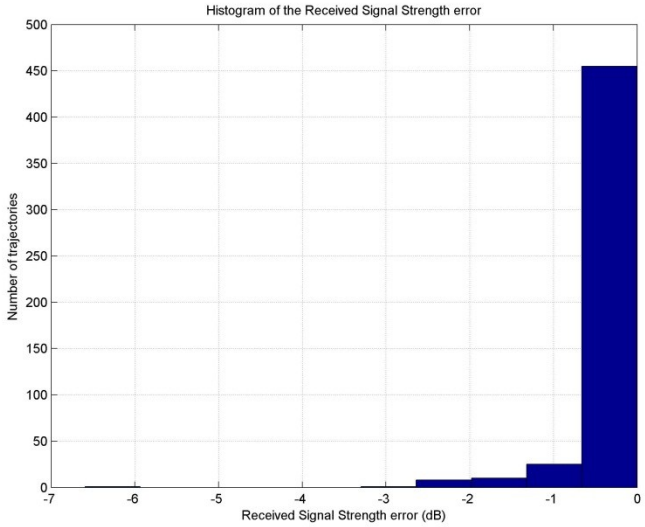

Figure 5.21 RSS errors histogram (turbulent trajectory).

Figure 5.22 shows the Time of convergence (TOC) for the 500 runs. The time of convergence is the time period consumed by the algorithm before producing its output. The average TOC is $445\,ms$ and there are 17 runs out of 500 that the algorithm was terminated by the time limit condition instead of the RSS improvement condition defined by t_d from equation (5.13). Figure 5.23 the histogram of the TOC is shown to give the reader a sense of how the TOC values are distributed on the time scale.

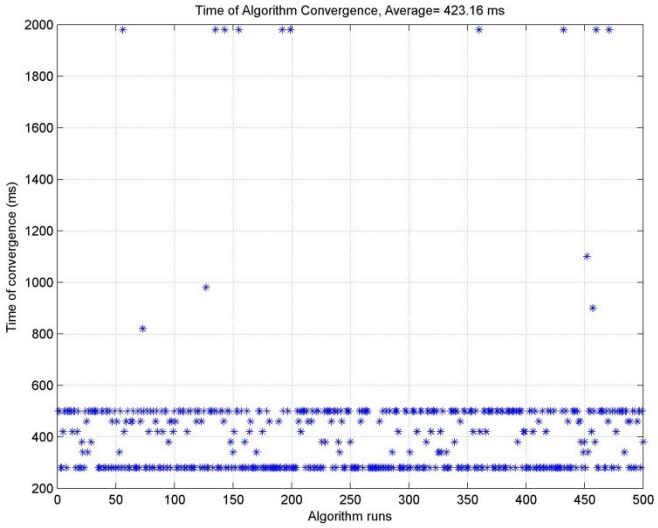

Figure 5.22 Time of convergence (turbulent trajectory).

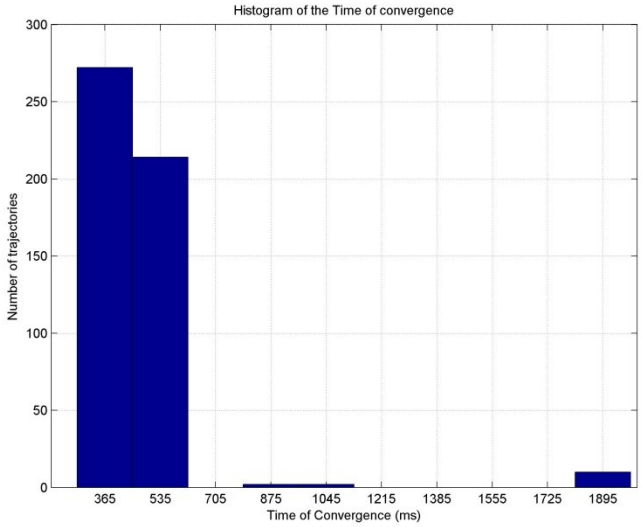

Figure 5.23 Histogram of TOC (turbulent trajectory).

In Figure 5.24 the errors in estimating the beam steering angle θ_b is shown. Notice that the error values have an average value of 8°. This is relatively low compared with the average error value of ϕ_b showed in Figure 5.25 which is 41.6°. This can be easily explained by referring to the antenna array chapter and recalling the fact that $HPBW_\theta$ is

usually smaller than $HPBW_{\phi}$. This will result in higher localization accuracy in the θ axis and a less localization accuracy in the ϕ axis.

By comparing the results of the smooth trajectories to the turbulent trajectories, we can notice that the error in ϕ_b decreased from $72.5°$ in the smooth to $41.6°$ in the turbulent. This can be explained by the fact that for a turbulent trajectory, the aircraft's rolling angle will have a relatively high average value during most of the flight time. This will lead to a less possibility of having the secondary beam instead of the primary pointing at the maximum RSS. Thus fewer values will be around $160°$ compared to the smooth trajectory. This is noticed clearly by comparing Figure 5.17 to Figure 5.25. Consequently the average error value decreased.

By comparing the errors in θ_b of the smooth trajectories to the turbulent trajectories, we can notice that the error in θ_b increased from $3.5°$ in the smooth to $8°$ in the turbulent. In the turbulent trajectories the average rolling angle increases, and consequently the actual average θ_b values will increase, which will result in a higher $HPBW_{\theta}$ on average. Thus, estimation errors will increase accordingly. In order to visualize this situation we must refer to the $HPBW_{\theta}$ contour plot in Figure 3.22.

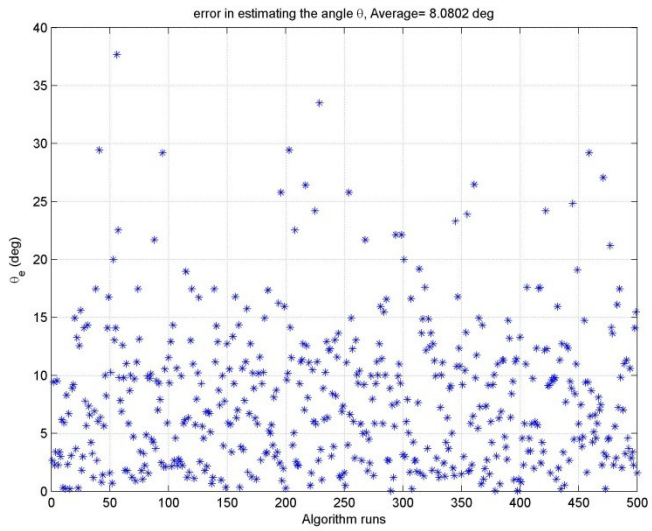

Figure 5.24 Errors in estimating θ_b (turbulent trajectory).

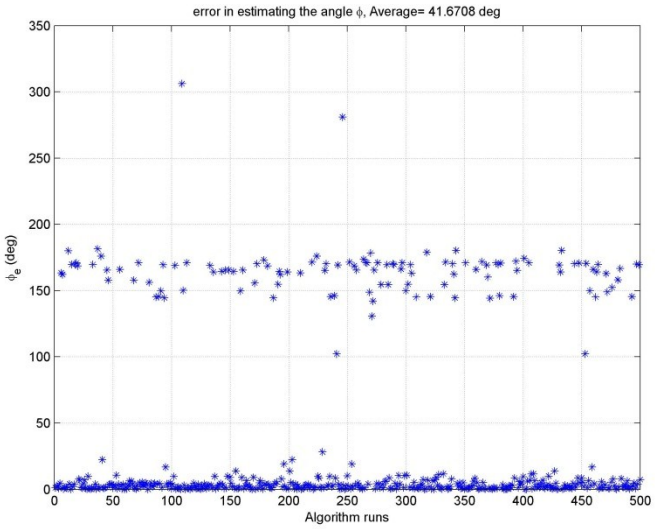

Figure 5.25 Errors in estimating ϕ_b (turbulent trajectory).

5.6 Differential evolution

5.6.1 Background

Around 1996, Rainer Storn and Kenneth Price introduced the *Differential Evolution* as an optimization technique [50]. Kenneth suggested to use vector differences for perturbing the vector population during his work to solve a problem called *Chebychev Polynomial*

fitting problem. Then, both made several improvements, until the DE was successfully formulated and introduced in [51].

The differential evolution (DE) algorithm is an evolutionary optimization technique. It is characterized by its simplicity, robustness, fast convergence, and small number of control variables.

DE requires a population of candidate solutions X_n, these solutions are grouped in what we call a generation of solutions G_i. Each solution X_n is a vector composed of a number of parameters that indicates the problem's dimensionality. In our case we only have two parameters in each vector (θ_b, ϕ_b).

$$X_n^i = [\, \theta_b \;\; \varphi_b \,] \tag{5.22}$$

$$G^i = [X_1^i \;\; X_2^i \;\; X_3^i \;\; \dots \;\; X_n^i \;\; \dots \;\; X_{NP}^i] \tag{5.23}$$

Note that NP is the number of solutions in each generation. The core of any evolutionary algorithm is the method of producing the offspring vectors (children vectors). In DE all individual solutions in a given generation are used to generate a trail solution. This is done by processing the solution through mutation, recombination, and selection operators. Then the best solution of the two compared is picked to be moved to the next generation as an offspring. The DE algorithm has been compared to the particle swarm algorithm in [52], where the author proved that DE is better in performance than the particle swarm. Keeping in mind that DE requires less number of objective function evaluations compared to particle swarm according to [53].

Figure 5.26 shows the flowchart of the main procedures for executing the DE algorithm. Each of the main blocks in the algorithm will be explained briefly.

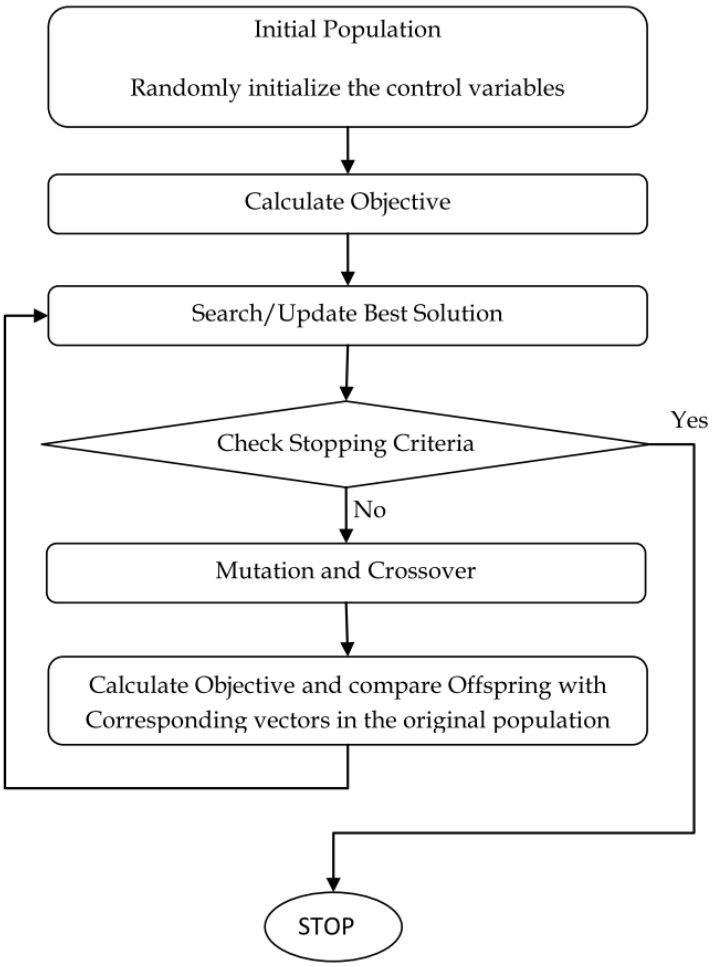

Figure 5.26 differential evolution flow chart.

a) Initialization

As a preparation stage, there are a number of factors that should be specified before proceeding.

- ❖ Fitness function: RSS value for the given solution
- ❖ Problem dimensionality: 2 (θ_b and φ_b).
- ❖ Problem constraints: The limits defined for (θ_b, ϕ_b)
- ❖ Population size "NS": 20
- ❖ Number of iterations: 5

- ❖ Mutation factor "F": 0.1
- ❖ Cross over factor "CR": 0.9

The first step is to generate the first generation. This is done by randomly choosing 10 solutions in the search space. Equations (5.24) and (5.25) present the model used for this random selection process for both parameters.

$$\theta_n^i = \theta_{min} + rand * (\theta_{max} - \theta_{min}) \qquad \forall n \in [1 \dots 10], i = 1 \tag{5.24}$$

$$\varphi_n^i = \varphi_{min} + rand * (\varphi_{max} - \varphi_{min}) \qquad \forall n \in [1 \dots 10], i = 1 \tag{5.25}$$

Where $rand$ is a uniformly distributed random number whose value lies in the interval [0,1].

b) Fitness evaluation

Evaluating the fitness of each solution is done by conducting an RSS measurement for this (θ_b, ϕ_b). Then all the fitness values will be used to compare between the solutions and choose the best one, which is the one with the maximum measured RSS value.

c) Mutation

This is considered as the first step towards creating a new generation of solutions from a previous generation. Each individual vector (solution) is mutated to produce another vector. This process can be performed using different mathematical models [51]. The model we used for this algorithm is

$$X_n^{im} = X_{best}^i + F * (X_{r_1}^i - X_{r_2}^i) \tag{5.26}$$

Where F is the mutation factor, $X_{r_1}^i$ is a randomly selected vector from the generation i, $X_{r_2}^i$ is another randomly selected vector from the same generation i, X_{best}^i is the best vector in the generation i (the one with the highest RSS value), and finally X_n^{im} is the mutant vector.

The mutation factor F takes a value between [0, 1]. The assigned value is usually achieved through trial and error. Note that as F approaches 0, the mutated vector will approach the original one and as F approaches 1, the mutated vector will be the most different from the original one.

d) Crossover

After mutation, the crossover process is performed to further perturb the trail solution in order to enhance the diversity of the generated population. In this process a crossover factor CR is defined as a numerical value lying in the interval $[0,1]$. This factor is assigned a value in the initialization stage and it retains this value throughout the algorithm execution. For each vector we already have two versions of it; the original vector X_n^i and the mutant vector X_n^{im}, a uniformly distributed random number $rand$ is generated with a value between 0 and 1. Then this random number is compared with the crossover factor. Each parameter in the vector is copied from either the original or the mutant vector to the new offspring vector according to this comparison. For each parameter in the trail vector, If $(rand \leq CR)$ then the parameter is taken from the mutant vector. And if $(rand > CR)$ the parameter is taken from the original vector.

e) Selection

After generating all the trail vectors, we compare each trail vector with its corresponding original vector and then choose the better one to be moved to the next generation. This selection process is performed according to the fitness function (the higher RSS value).

f) Stopping criteria

For any newly produced generation, the global best solution is updated, also the stopping criteria is checked. Usually the stopping criterion is the maximum number of generations which is defined in the initialization stage.

5.6.2 Simulation Results

In this section the simulation results of the DE algorithm will be presented, and its performance after simulating 500 flight trajectories is evaluated. We will repeat it twice, one time for a smooth nearly straight flight trajectory and another for a turbulent flight trajectory with a lot of tight maneuvers. This is done to assess the algorithm performance in all flying circumstances. Notice that the TOC of the DE algorithm is fixed at $500\ ms$ by setting the population size to 20 and letting the populations evolve for only 5 generations. Since one value consumes $5\ ms$ [22], therefore $20 \times 5 \times 5\ ms = 500\ ms$.

5.6.2.1 Smooth trajectory Results

In Figure 5.10 a top view of a smooth trajectory is shown which is the same as the one used to test the EP algorithm. One can observe that the trajectory is relatively smooth. We will use the same trajectory to assess the performance of the DE algorithm.

Figure 5.27 shows the errors in the RSS values. This is the difference between the maximum achievable RSS and the RSS achieved by the DE algorithm. Notice that the average error value is $-0.99\ dB$. In Figure 5.28 a histogram of the RSS error values are shown to provide the reader with the sense of how the values are distributed among the signal strength error scale. One can easily observe that the performance of the EP algorithm is better than the DE algorithm in the smooth trajectory. By comparing the

average RSS errors of the EP algorithm to the DE algorithm, we notice that it increased from $-0.24\ dB$ in the EP algorithm, to $-0.99\ dB$ in the DE algorithm which is almost 4 times the value of the EP algorithm. This is due to the randomness nature of the DE algorithm.

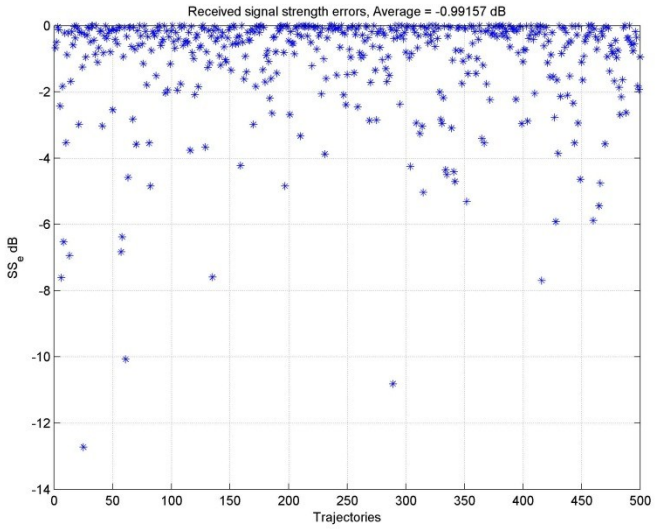

Figure 5.27 Errors in the RSS obtained by the DE algorithm through all the 500 trajectories.

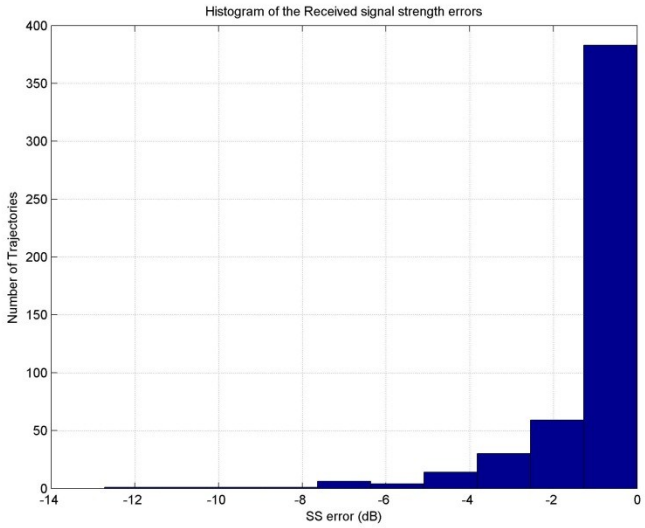

Figure 5.28 RSS errors histogram (smooth trajectory).

In Figure 5.29 the errors in estimating the beam steering angle θ_b are shown. Notice that the error values have an average value of 23.7°. In Figure 5.30 the error in estimating ϕ_b

is shown where the average value is $59°$. This can be explained by referring to the antenna array chapter and recalling the fact that $HPBW_\theta$ is usually smaller than $HPBW_\phi$. This will result in higher localization accuracy in the θ axis and a less localization accuracy in the ϕ axis.

Notice that in Figure 5.30 the error pattern is scattered compared to those produce by the EP algorithm in Figure 5.17. This is due to the randomness nature of evolutionary algorithms. In other words each run will produce totally different results yet all the runs will maintain the same statistical characteristics.

When we compare the DE algorithm to the EP algorithm in terms of estimation errors of θ_b, we notice that the value for the EP was $3.5°$ and it increased for the DE to $23.7°$. This is due to the randomness nature of the DE algorithm where each generation is produced randomly from its ancestor. Thus the trend in updating θ_b values will not be necessarily improving, and that will produce large variance in the error values, which is the case in Figure 5.29.

The average ϕ_b errors for the EP were $72.5°$ compared to $59.1°$ for the DE. This reduction is due to the randomness nature of the DE algorithm. By observing the error trends in Figure 5.30 we can see that the values are not gathered around $160°$ as it was the case in Figure 5.17, instead they are disturbed and this is why the average ϕ_b error value decreased in the DE algorithm compared to the EP algorithm.

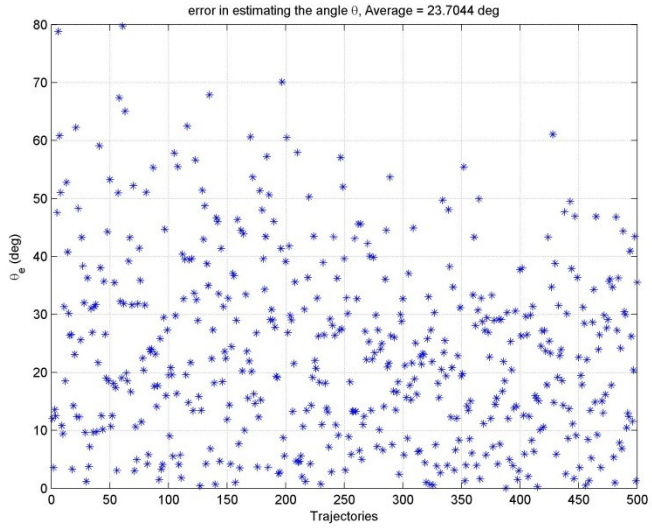

Figure 5.29 Errors in estimating θ_b (smooth trajectory).

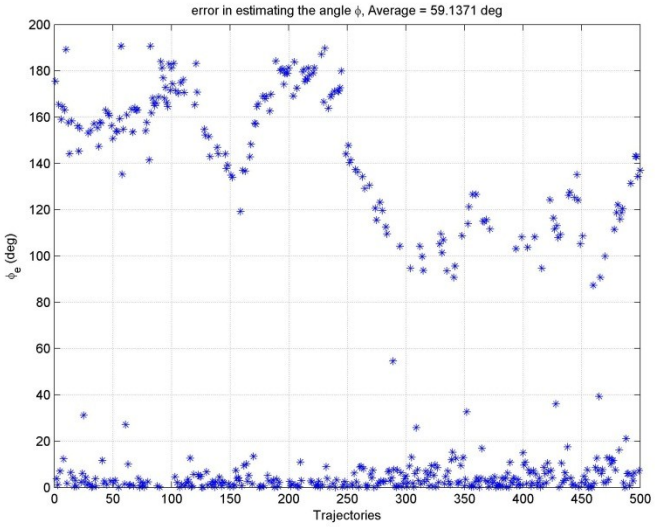

Figure 5.30 Errors in estimating ϕ_b (smooth trajectory).

5.6.2.2 Turbulent trajectory Results

In Figure 5.18 a top view of a turbulent trajectory is shown which is the same one used to test the EP algorithm. One can observe that the trajectory has a lot of tight turns. This trajectory will be used to assess the performance of the DE algorithm.

Figure 5.31 shows the errors in the RSS. This is the difference between the maximum achievable RSS and the RSS achieved by the DE algorithm. Notice that the average error value is $-0.96\,dB$ compared to $-0.245\,dB$ from the EP algorithm. In Figure 5.32 a histogram of the RSS error values are showed to provide the reader with the sense of how the values are distributed among the signal strength error scale. One can easily observe that the performance of the EP algorithm is better than the DE algorithm in the turbulent trajectory by just comparing between Figure 5.21 and Figure 5.32, where the histogram of the DE shows higher error variance compared to the EP algorithm, indicating less accuracy in achieving the maximum RSS.

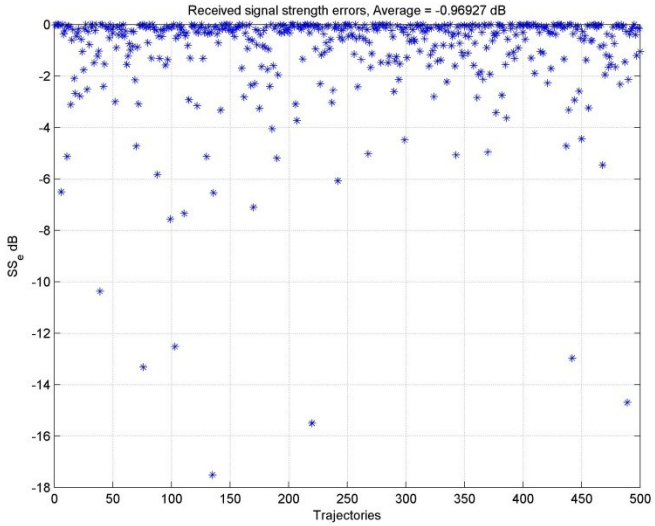

Figure 5.31 Errors in the RSS obtained by the DE algorithm through all the 500 trajectories.

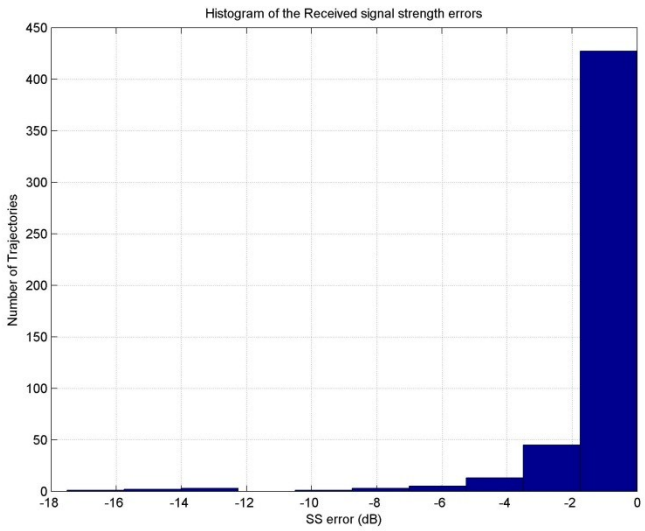

Figure 5.32 RSS errors histogram (turbulent trajectory).

In Figure 5.33 the errors in estimating the beam steering angle θ_b are shown. Notice that the error values have an average value of $21.4°$. In Figure 5.34 the error in estimating ϕ_b is shown where the average value is $61.5°$. This is explained by referring to the antenna array chapter and recalling the fact that $HPBW_\theta$ is usually smaller than $HPBW_\phi$. This will result in higher localization accuracy in the θ axis and a less localization accuracy in the ϕ axis.

Notice that in Figure 5.34 the error values are scattered compared to those produced by the EP algorithm in Figure 5.25 where the values are gathered around $160°$. This is due to the randomness nature of evolutionary algorithms. In other words each run will produce totally different results yet all the runs will maintain the same statistical characteristics.

The randomness nature of the DE algorithm is also the reason for getting almost the same performance with the turbulent $(-0.97\ dB)$ compared to the smooth trajectory $(-0.99\ dB)$. This is also the case for the EP algorithm where its average RSS error in the

smooth trajectory was $-0.244\ dB$, and in the turbulent trajectory was $-0.245\ dB$, but in the EP algorithm this observation is explained by the randomness of the trajectories used to simulate the algorithm. In other words, each time the algorithm is executed, the trajectory used is picked randomly from the 500 trajectories population. This will give a better statistical sense of the algorithm performance.

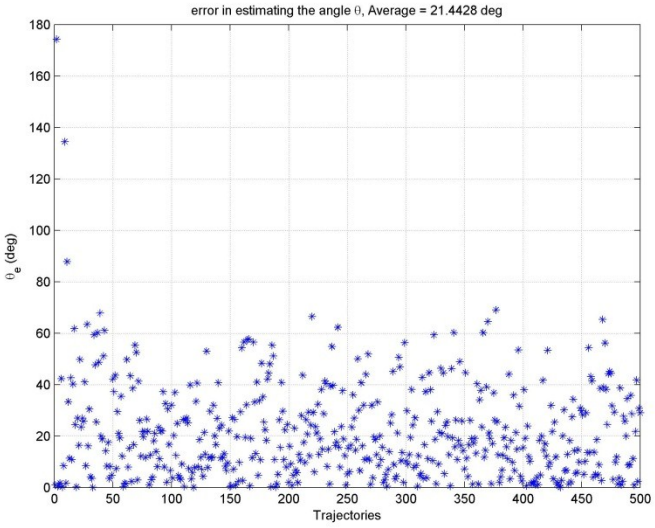

Figure 5.33 Errors in estimating θ_b (turbulent trajectory).

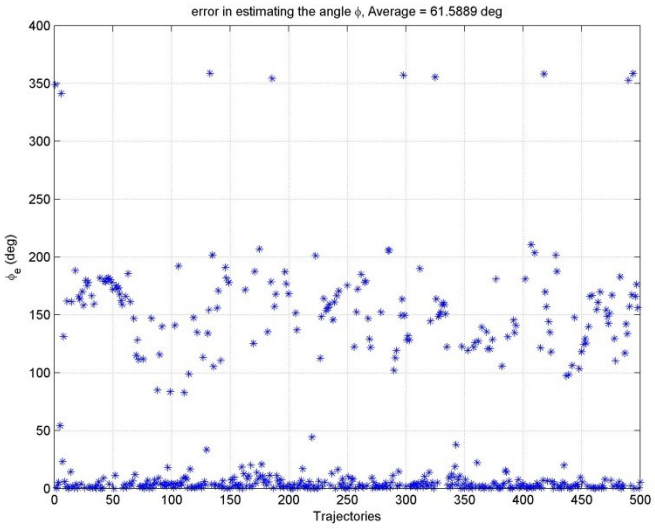

Figure 5.34 Errors in estimating ϕ_b (turbulent trajectory).

5.7 Conclusions

In this chapter we showed simulation results for the proposed EP and the DE algorithm. This was done for two trajectories, a smooth trajectory with a nearly straight path, and a turbulent trajectory that has a lot of tight turns. Results are compared in Table 5.1. The EP Algorithm showed better results in terms of RSS beam miss-alignment errors. The average error in the EP algorithm was $-0.16\ dB$ and $-0.27\ dB$ for the smooth and the turbulent trajectories respectively. The average error in the DE algorithm was $-0.99\ dB$ and $-0.96\ dB$ for the smooth and the turbulent trajectories respectively.

The lower performance of the DE in terms of the average RSS error can be due to the fact that we have a dynamic search space where by the time we create a new generation in the DE algorithm, the global best would have changed its place; also the fitness function geometry in the search space would have changed. This can be resembled as a predator chasing its prey and trying to catch it but the prey is faster than the predator, therefore it will never be able to catch it unless by a random chance.

Both Algorithms (EP and DE) don't provide accurate estimations for the steering angles, where in the EP algorithm the average estimation error in θ_b and ϕ_b for the smooth trajectory was $3.5°$ and $72.5°$ respectively. In the DE algorithm the average estimation error in θ_b and ϕ_b for the smooth trajectory was $23.7°$ and $59.1°$ respectively. These errors don't have any significance in the system performance, because we are only interested to maximize the RSS. This is achieved because the beam is relatively wide, which preserve the system performance even with errors in the steering angles as long as the actual maximum RSS falls inside the footprint of the beam.

Table 5.1 Simulation results comparison.

	Average error		Time of Convergence	
Trajectory Type	**Smooth**	**Turbulent**	**Smooth**	**Turbulent**
EP	$-0.244\ dB$	$-0.245\ dB$	$431\ ms$	$423\ ms$
DE	$-0.991\ dB$	$-0.996\ dB$	$500\ ms$	$500\ ms$

Chapter 6: Tracking Routine

In this chapter we will discuss the tracking routine employed to keep the antenna beam steered towards the maximum RSS as the UAV moves. This routine depends on collecting RSS readings as well. Thus the XBee module [22] must be set to work in the receiving mode. Since our final goal is to improve the communication link which will carry the data from the UAV to the ground station, therefore most of the time the XBee module will work as a transmitter. During the execution of the EP algorithm, no data will be allowed to be transmitted to the ground station from the UAV, and that is why we needed to look into the TOC of the EP algorithm and make sure to minimize it as much as possible.

The XBee module [22] supports a number of communications protocols which include the IEEE 802.15.4 protocol [54]. The module specification sheet [22] clearly states that reliable packet delivery is based on Retries/Acknowledgements schemes. This will enable us to easily pause the transmission until we execute the EP algorithm which will take nearly $450\ ms$, then proceed with the data packets transmission.

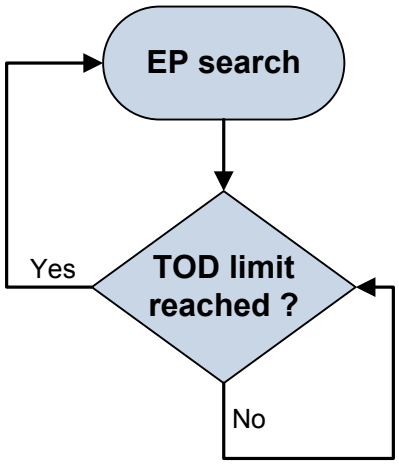

Figure 6.1 Tracking routine flow chart

In Figure 6.1 the flow chart of the tracking routine is shown. It is very simple to execute on a hardware platform because it only requires an independent timer which is available on almost any off the shelf microcontroller. The EP algorithm will produce the estimates of (θ_b, ϕ_b) and then they will be accounted as the beam direction while we are transmitting the data packets, yet we must keep in mind that the UAV is constantly in motion which will make these estimates outdated in a very short period of time. We wanted to assess how long it will take until the estimates become far from the actual values and cause signal degradation due to beam misalignment. This is done via simulations where we executed the EP algorithm then we kept measuring the RSS at the same estimated (θ_b, ϕ_b) values as the aircraft moves until we recorded a $-3\ dB$ degradation from the maximum achievable RSS, then we recorded the time passed till we observed this event. This time period is called time of degradation (TOD). In other words TOD is the time period after which the estimated beam steering angles (θ_b, ϕ_b) will be far from the actual values such that they cause a signal degradation of $-3\ dB$. The TOD

will be the time limit after which another EP search session will be conducted to update the estimates of the beam steering angles (θ_b, ϕ_b).

A simulation to evaluate the TOD has been performed using the same two trajectories used for testing the EP and the DE algorithms. They are the smooth and the turbulent trajectories. Accordingly, another simulation was performed to test the tracking abilities of this routine. The results are shown in section 6.1.

6.1 Simulation Results

In this section we will present the simulation results for the TOD evaluations for 14 different trajectories. This was done two times, one for the smooth trajectory shown in Figure 5.10, and the other for the turbulent trajectory shown in Figure 5.18. It is worth mentioning that to perform this simulation we needed a trajectory fragment of long duration. The ones we used for the search algorithms testing were 3 seconds long while the ones we needed here were 120 seconds. This is the reason for the reduced number of trajectory fragments investigated (14 instead of 500).

In Figure 6.2 the TOD for the smooth trajectory is shown. The average value is 45.5 seconds. This is considered very good, because it means we can keep the XBee module in the transmitting mode for 45.5 seconds before needing to update the estimates of (θ_b, ϕ_b). Thus the data throughput will not be much affected. Even if we choose the worst case ($\sim 15\ s$) the time duration is considered good for transmission.

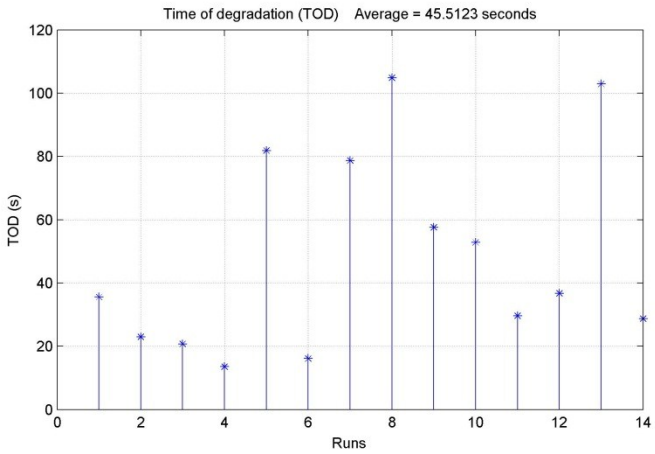

Figure 6.2 TOD evaluation for 14 different trajectories (smooth).

In Figure 6.3 the TOD for the turbulent trajectory is shown. The average value is 10.7 seconds. This is almost one fourth of the value for the smooth trajectory. This can be easily explained by the fact that in the turbulent trajectory, the aircraft changes its attitude very often causing faster changes in the actual (θ_b, ϕ_b) values. Thus the estimates will not last very long and the will need to be updated sooner. The worst case TOD is 2 seconds.

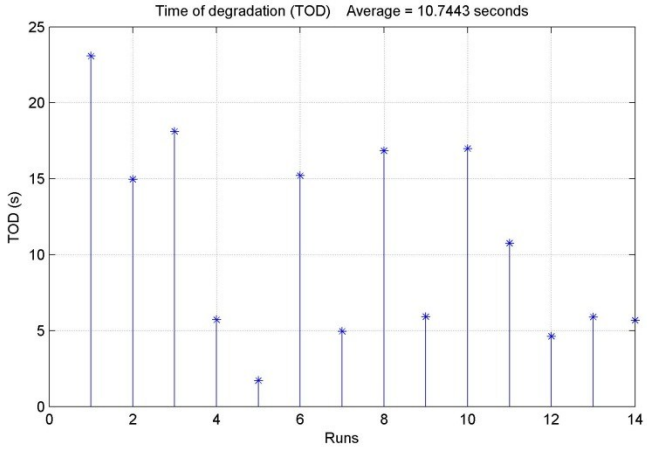

Figure 6.3 TOD evaluation for 14 different trajectories (turbulent).

In Figure 6.4, the tracking results of the smooth trajectory are shown. These results were obtained using a TOD limit of 10 seconds. In other words every 10 seconds a new EP search session is conducted to update (θ_b, ϕ_b). Keep in mind that this is done for a trajectory that doesn't have any sudden maneuvers. Thus the 10 seconds period gives reasonable tracking and will not affect the data throughput significantly. In other words the data transmission will be interrupted every 10 seconds to conduct the EP search session which takes around 450 ms.

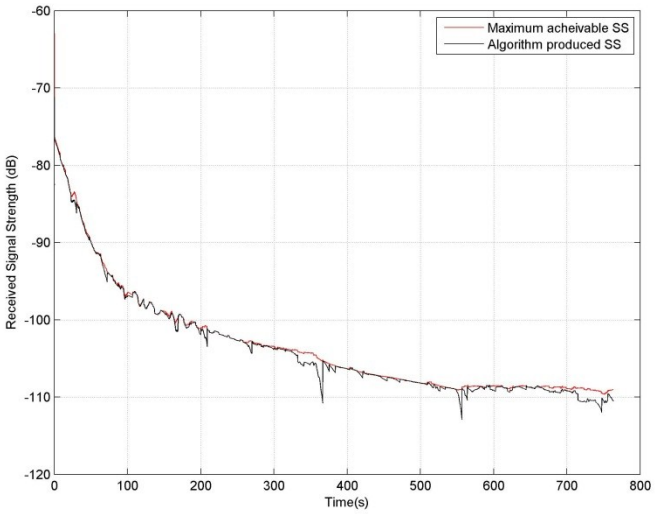

Figure 6.4 Tracking maximum RSS, (smooth trajectory, $TOD = 10s$).

In Figure 6.5 the tracking results for the turbulent trajectory are shown for a TOD limit of 10 seconds. Since this trajectory has a lot of sudden un expected maneuvers, therefore the track can be lost easily within a period of 10 seconds. Thus to improve the tracking we need to reduce the TOD limit.

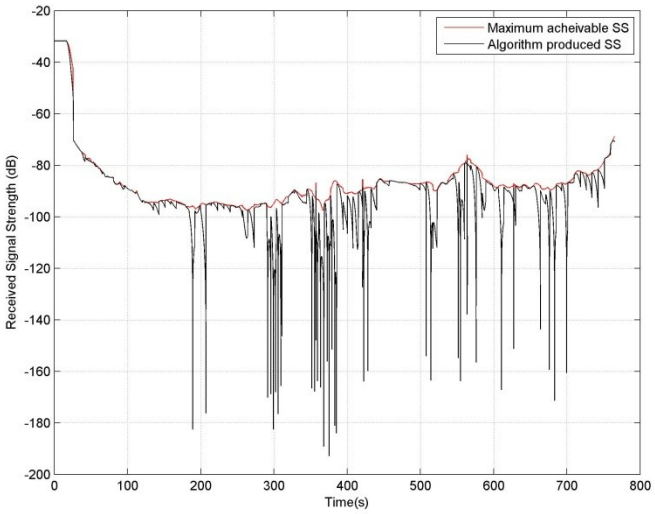

Figure 6.5 Tracking maximum RSS, (turbulent trajectory, $TOD = 10s$).

In Figure 6.6 the same turbulent trajectory was used, but the TOD limit was reduced to 5 seconds. Notice that the accuracy is improved because we now update our estimates every 5 seconds only instead of 10 seconds. Keep in mind that there is still some points at which the tracking routine fails to track the maximum RSS.

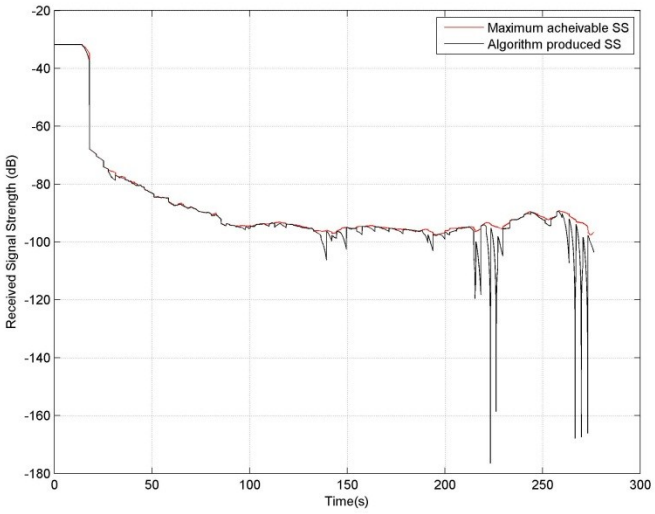

Figure 6.6 Tracking maximum RSS, (turbulent trajectory, $TOD = 5s$).

In Figure 6.7 the same turbulent trajectory was used, but even more reduction in the TOD limit. It value now is only 3 seconds. One can easily see that the tracking accuracy has improved where now there is no significant signal degradation. But keep in mind that reducing the TOD limit to 3 seconds will accordingly reduce the data throughput, because in this case there are more interruptions in the transmission.

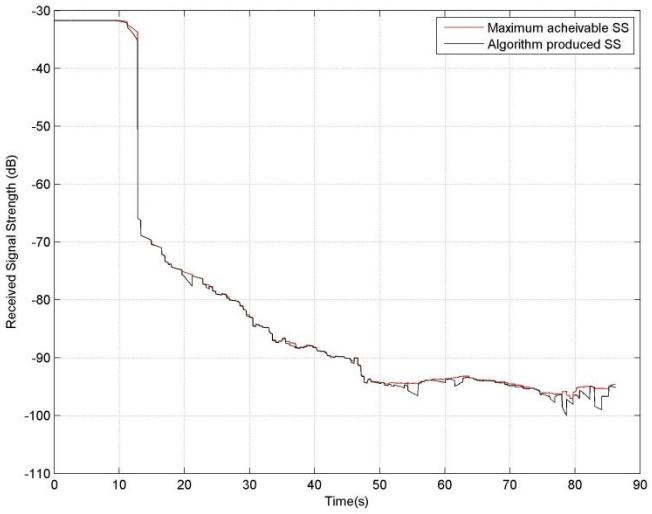

Figure 6.7 Tracking maximum RSS, (turbulent trajectory, $TOD = 3s$).

6.2 Conclusions

In this chapter we presented the tracking routine used to keep the beam steered towards the maximum RSS as the flying vehicle move around. For a smooth flight trajectory the tracking routine will not interrupt the data transmission frequently, where a successful tracking case was achieved with $TOD = 10s$. Thus good throughput is achievable. For a turbulent flight trajectory the tracking routine gave a successful tracking case with $TOD = 3s$. This will require frequent data transmission interruptions to realign the antenna beam with the maximum RSS, Thus data throughput will be reduced.

Chapter 7: Conclusions and Future work

In this chapter we will present our general thesis conclusions. Our vision for possible future work is also discussed in order to lay the ground for any possible improvements in the future.

7.1 Thesis Conclusions

In this work we presented an integrated algorithm that consists of a search algorithm that identifies the direction of the maximum received signal strength under the wing structure of a flying UAV, a tracking routine, a beam steering algorithm, and the design of a planar antenna array for communication link enhancement of the flying UAV.

A planar patch antenna array that is to be embedded inside the wing structure of a mini-UAV is designed and fabricated. Two array configurations were explored, one with 14 elements (7×2) and the other with 12 elements (6×2). Both arrays were characterized in terms of radiation pattern, HPBW and pointing gain versus all possible steering angles (θ_b, ϕ_b).

Scattering parameters were measured to quantify the resonant frequency and the isolation between adjacent elements. The antenna array resonated at $2.48\,GHz$ with minimum $-10\,dB$ bandwidth of $30\,MHz$ and the adjacent elements coupling did not exceed $-25\,dB$.

A virtual aircraft trajectory was generated using two methods. The randomly generated trajectory and a flight simulator based trajectory. A feasibility test was conducted for both trajectories using a dynamic model to prove that the generated trajectory is valid for

UAVs. The randomly generated trajectory failed the feasibility test while the flight simulator based trajectory passed the feasibility test. Thus it was used in our proposed searching and tracking algorithm.

The elliptical Peeking (EP) algorithm has been proposed and tested extensively for a smooth flight trajectory as well as a turbulent one. It showed good performance where it was able to find the maximum RSS within $450\ ms$ on average. The EP algorithm was compared to the differential evolution (DE) algorithm in order to benchmark the performance of the proposed algorithm (EP). The EP Algorithm showed better results in terms of RSS beam miss-alignment errors. The average error in the EP algorithm was $-0.16\ dB$ and $-0.27\ dB$ for the smooth and the turbulent trajectory respectively. The average error in the DE algorithm was $-0.99\ dB$ and $-0.96\ dB$ for the smooth and the turbulent trajectory respectively.

The tracking routine was integrated with search algorithm and tested. It showed acceptable performance in the turbulent trajectory, but it showed better performance in the smooth trajectory. Data transmission was interrupted every 10 seconds in the smooth trajectory to re-align the beam, while for the turbulent trajectory data transmission was interrupted every 3 seconds, this will decrease the overall data throughput.

7.2 Future work

Some future work that can be built upon the results in this thesis:

1. The system can be modified in order to be used for navigation and localization, by increasing the size of the antenna array. Thus a narrower beam can be obtained.

This will cause more delay in the EP algorithm, and the system complexity will increase, but we will achieve much better accuracies in the steering angles.

2. Optimizing reduction factors ($f_n \; and \; f_r$) in the EP algorithm for the best performance, by using a well-known optimization method instead of settling with values obtained from trial and error.
3. For the tracking routine, we need to assess the performance in terms of data throughput. This can be achieved by calculating the time slots allowed for transmission in each trajectory type and accordingly relate it to the amount of throughput achievable within these time slots. To do that we will need to study the communication protocol [54] used by the XBee module [22].
4. Downloading the EP algorithm on a hardware platform in order to assess its execution time and memory demands is another important addition.
5. Field testing of the system using the mini-Telemaster [23] UAV equipped with the fabricated antenna array and the XBee module [22] is essential to test the algorithm performance in a real time environment with hardware implementation.

Appendix (A)

This appendix presents a tutorial to show the main steps when using *FlightGear* [37] to simulate an aircraft flight and record the flight log in order to be used for trajectory extraction and further processing.

It is recommended to use a very good graphics card for running *FlightGear*. After downloading the flight gear source and setting up the software, open the simulator and you will find the aircraft selection menu shown in Figure 8.1. In this menu you can select the aircraft which you will fly. From my own experience the easiest one for beginners is the dragonfly. Then after getting some flying skills you can move to a more advanced aircraft such as the Cessna 172.

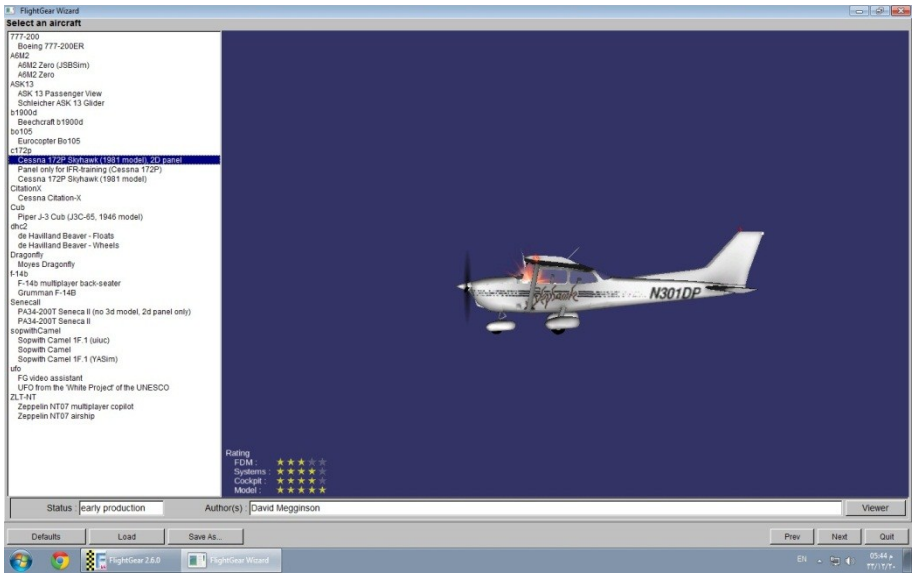

Figure 8.1 *FlightGear* aircraft selection menu.

After selecting your aircraft, select the airport from which you will take off. In Figure 8.2 the airport selection menu is shown.

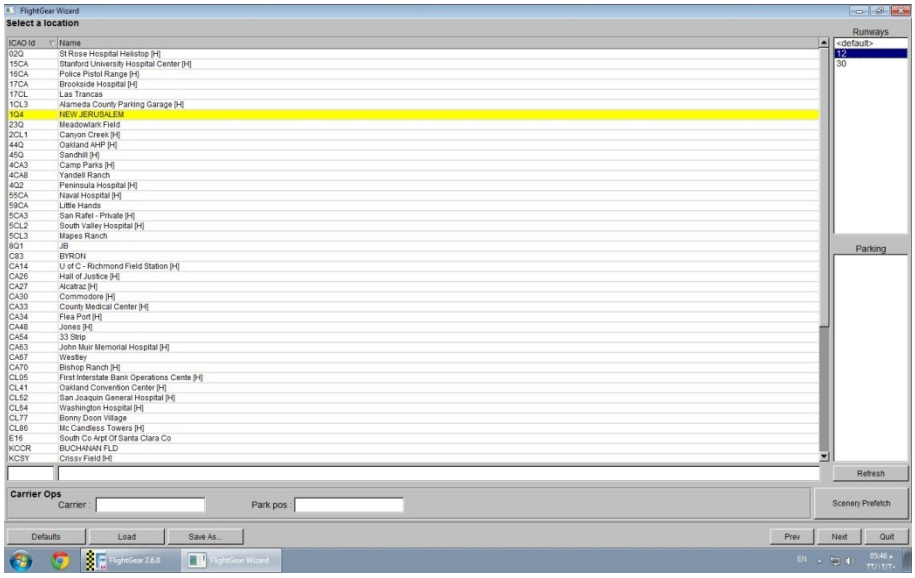

Figure 8.2 *FlightGear* airport selection menu

The final step before loading the simulation environment is setting the simulation settings in the wizard shown in Figure 8.3. This wizard allows you to set the resolution, the time of the day, the graphical objects and also you can control some advanced features by clicking the advanced button in the lower right corner.

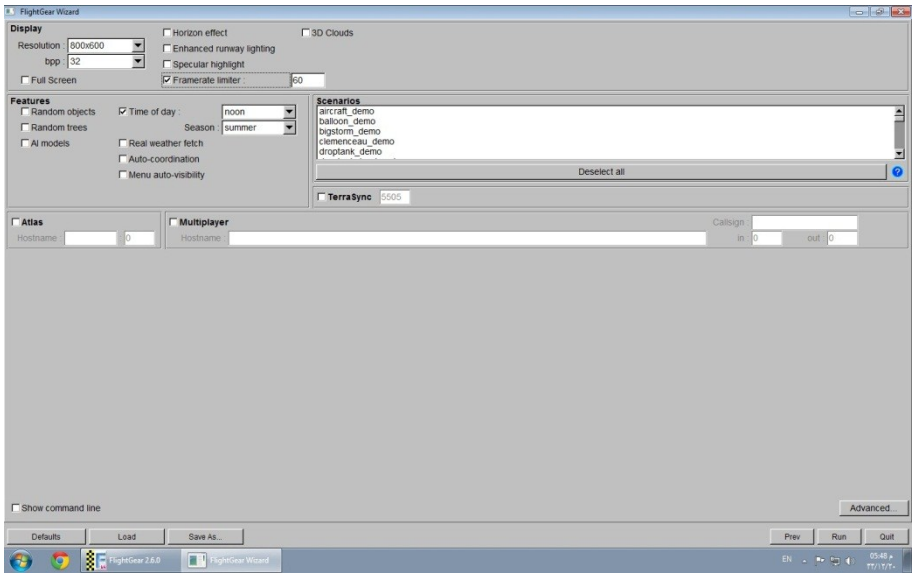

Figure 8.3 *FlightGear* simulation settings.

After hitting the run button, the simulator will start loading and you will see the screen in Figure 8.4.

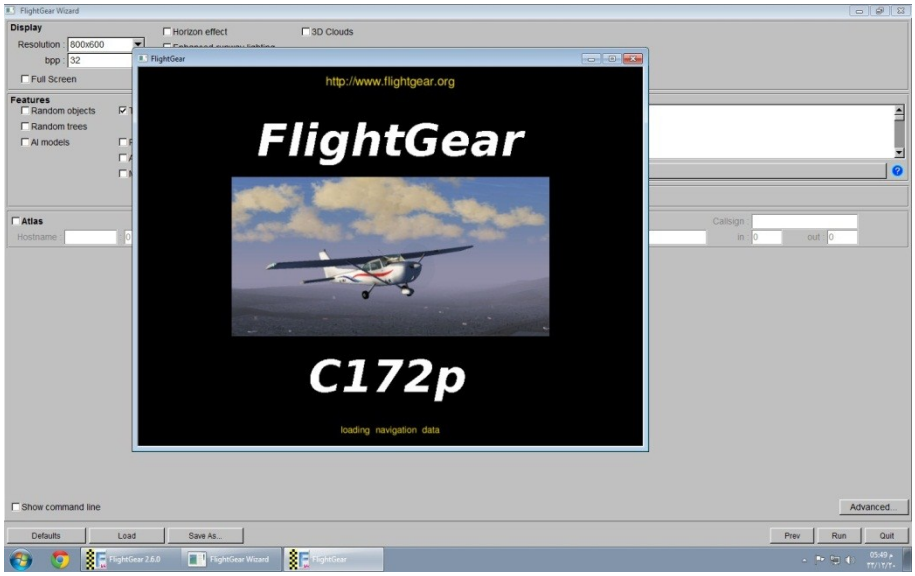

Figure 8.4 *FlightGear* loading screen.

Once the loading is finished, the simulator will open the environment showing the aircraft such as the screen in Figure 8.5.

Figure 8.5 *FlightGear* simulation environment loaded.

The flight logging process must be set now by going to the *Debug* menu, then hitting the *Logging* item within the menu.

Figure 8.6 *FlightGear* Logging Procedures.

Once clicked, the logging window will appear. Set it exactly as it appears in Figure 8.7. This way you will log 7 parameters, the longitude, latitude, altitude, roll, pitch, heading (yaw), and the air speed. More parameters can be logged by just writing their corresponding command in this wizard.

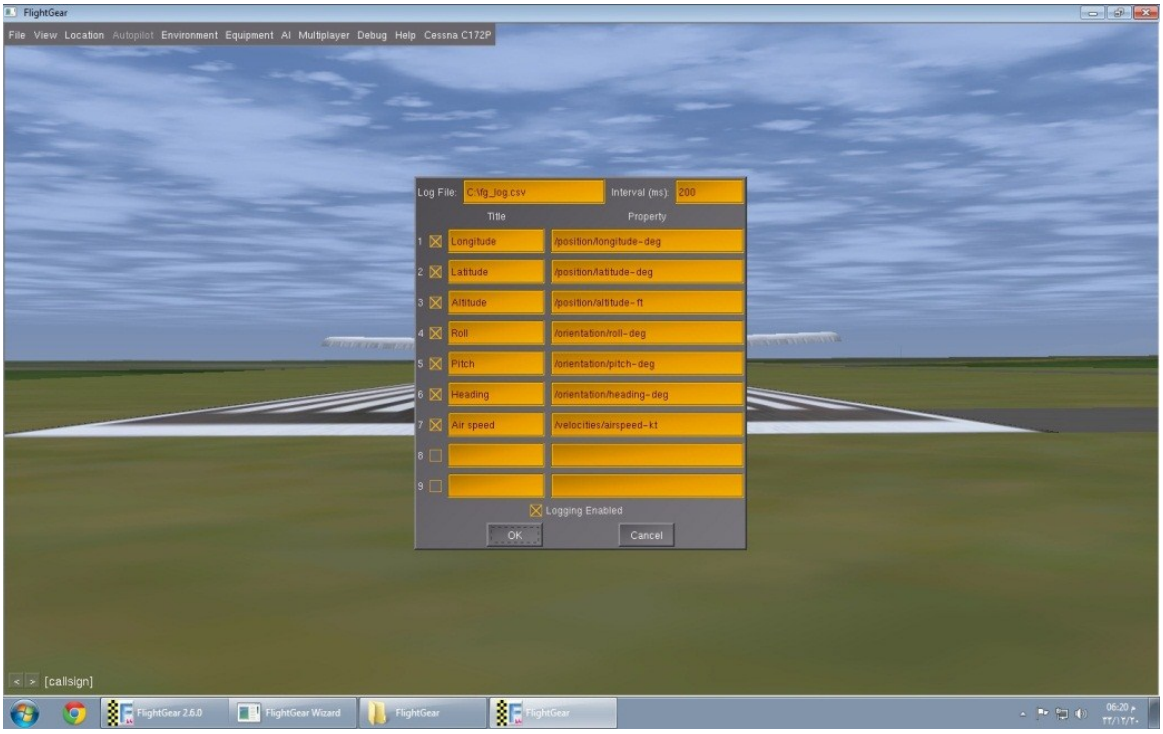

Figure 8.7 *FlightGear* Logging wizard

Then hit *ok* and start flying your aircraft. Once you exit *FlightGear,* the file will be automatically saved in the destined location in a ".csv" format.

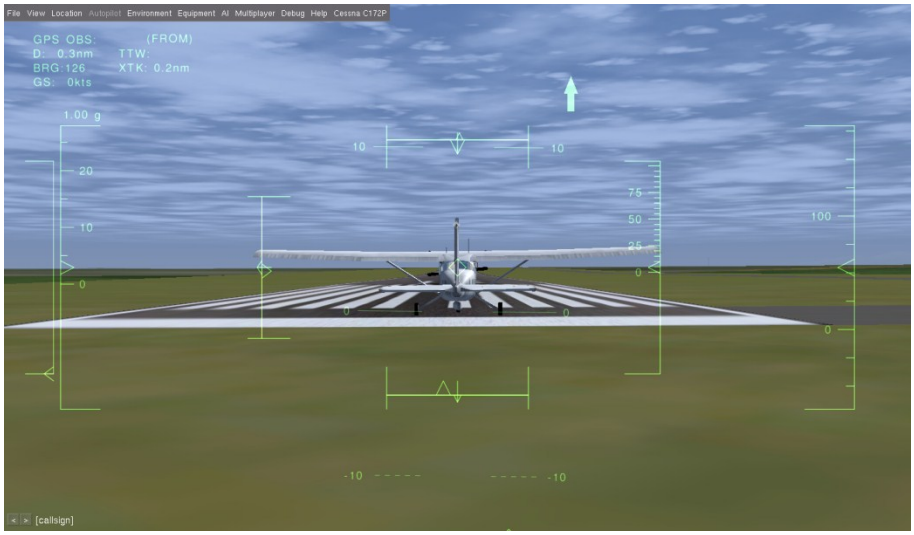

Figure 8.8 *FlightGear* take off position.

Note that you will take a while before being able to successfully fly and land the aircraft, because let's not forget that this is a flight simulator that can be used to train real pilots. Also keep in mind that each aircraft has its own properties which determine its take off velocity, its stalling speed, .etc. Thus a separate training must be conducted for each aircraft, but off course the basics are common for all aircrafts.

Appendix (B)

The virtual trajectory samples generated from the virtual trajectory generator are referred to an inertial frame centered at the ground station. To be able to view the situation from the vehicle's point of view, a new frame of reference will be considered which will be centered at the vehicles center of mass. All the trajectory samples will be transformed from the ground station centered frame to the vehicle centered frame. This transformation will be conducted upon all the six sample variables (X_u, Y_u, Z_u, r, p, y).

Figure 3.1 shows the UAV centered frame of reference, where the x-axis is directed along the vehicle's fuselage and pointing to its heading. The y-axis is along its wings and pointing for the right wing. The z-axis is accordingly pointing downwards. This reference frame is aligned this way to match the same reference frame used by [30] in calculating the radiation pattern of a planar array. This will simplify calculations when it comes to simulating the whole system.

The new coordinates will be denoted by a subscript "t", which is short for "ground transmitter". A total of six sample variables are required to be transformed. The transformation will start by rotating the frame by the three attitude angles and then introduces three translational vectors. Finally, three position coordinates will be obtained for the ground transmitter at each trajectory sample. In other word we will have a trajectory for the ground station motion as seen from the flying vehicle. Keep in mind that the ground station has a fixed position on the ground and it is not mobile.

- **Translation**

The translation of the 3 position variables is much simpler and easier than the rotation variables. All what needs to be done is to relate the axes in Figure 9.1 (a) to $X_u, Y_u, and\ Z_u$ in the x, y, z directions respectively. This gives equations (9.1), (9.2), and (9.3).

$$X_t = -X_u \tag{9.1}$$

$$Y_t = -Y_u \tag{9.2}$$

$$Z_t = -Z_u \tag{9.3}$$

Therefore, by inverting the signs of $X_u, Y_u, and\ Z_u$ we would have accounted for the position transformation required.

- **Rotation**

Rotating a Cartesian frame can be performed by applying the rotation matrix corresponding to the rotated axis. The three rotation matrices have to be created. Then we will apply them to rotate the original frame until it coincides with the new frame. Figure 9.1 shows the rotation action that is considered as a priory step before applying the attitude rotations.

For the rotation matrices stated in equations (9.4), (9.5), and (9.6), each of these basic vector rotations is assumed to appear in two conditions:

1- In a counter-clockwise direction when the axis around which they occur is pointing towards the observer.

2- The coordinate system is right-handed.

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{bmatrix} \tag{9.4}$$

$$R_y(\theta) = \begin{bmatrix} cos(\theta) & 0 & -sin(\theta) \\ 0 & 1 & 0 \\ sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \tag{9.5}$$

$$R_z(\theta) = \begin{bmatrix} cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{9.6}$$

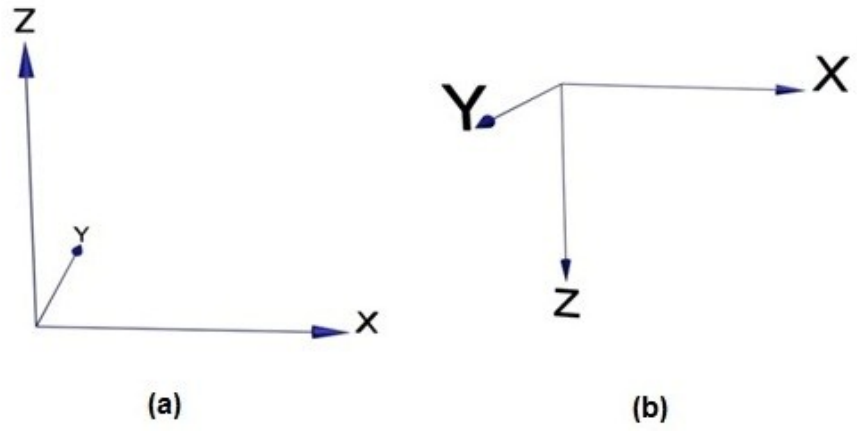

Figure 9.1 Rotation actions where (a) is the original frame, (b) is the rotation of (a) by 180° around X-axis.

In Figure 9.1 the frame is rotated by 180° around the x-axis. By substituting in equation (9.4) we can resemble this rotation by the matrix given by equation (9.7).

$$R_x(180°) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \tag{9.7}$$

Now we will consider the Yaw angle of the vehicle, where as it was defined earlier, it is the angle measured from the $+ve$ x-axis of the original frame to the vehicle's heading which is along the $+ve$ x-axis of the new frame. Therefore the Yaw is a rotation around the z-axis with the value $-Yaw$ because as we rotate, the direction of rotation is not

counter-clockwise as proposed by the assumption used to construct the three rotation matrices given by equations (9.4), (9.5), and (9.6).

$$R_z(-Yaw) = \begin{bmatrix} cos(Yaw) & sin(Yaw) & 0 \\ -sin(Yaw) & cos(Yaw) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{9.8}$$

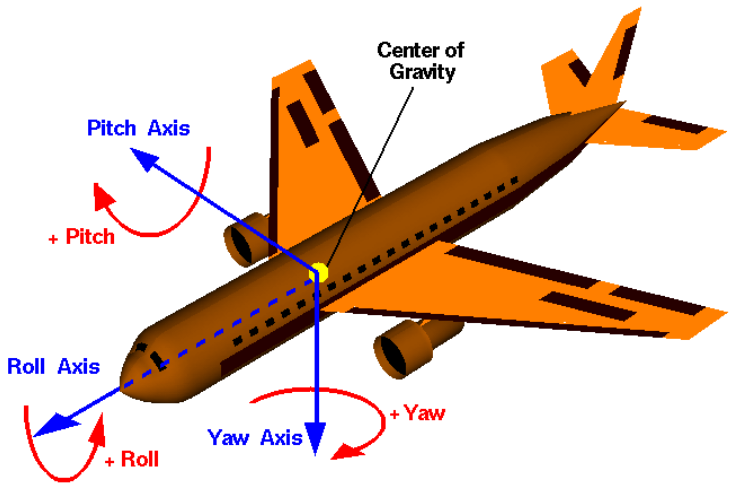

Figure 9.2 Graphical illustration for the attitude angles (roll, pitch and yaw) [55]

For the rolling angle r defined by Figure 9.2 the direction of rotation is counter-clockwise when the x-axis is pointing towards the observer. This definition coincides with the assumption of the given rotation matrices. Therefore the rolling angle will be used as it is to substitute in equation (9.4) which will give us equation (9.9).

$$R_x(r) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(r) & -sin(r) \\ 0 & sin(r) & cos(r) \end{bmatrix} \tag{9.9}$$

Finally for the pitch angle p defined by Figure 9.2 which is a rotation around the y-axis, its definition also matches our rotation matrices assumption. Therefore it will be used as it is to substitute in equation (9.5) which gives us equation (9.10).

$$R_y(p) = \begin{bmatrix} cos(p) & 0 & -sin(p) \\ 0 & 1 & 0 \\ sin(p) & 0 & cos(p) \end{bmatrix} \quad (9.10)$$

In order to deduce the final rotation matrix which will include all the above rotation actions we should multiply the matrices given by equations (9.7), (9.8), (9.9), and (9.10) together. This multiplication is stated in equation (9.11) which shows the relation between the original and the new frame.

$$[X_t \quad Y_t \quad Z_t] = [-X_u \quad -Y_u \quad -Z_u] \times [R_x(180°) \times R_z(-Yaw) \times R_x(r) \times R_y(p)] \quad (9.11)$$

By working out equation (9.11) we get

$$[X_t \quad Y_t \quad Z_t] = [-X_u \quad -Y_u \quad -Z_u] \times [TM]_{3\times3} \quad (9.12)$$

Where $[TM]_{3\times3}$ is the transformation matrix given by equation (9.12)

$$[TM]_{3\times3} = \begin{bmatrix} \cos(y)\cos(p) - \sin(y)\sin(r)\sin(p) & sin(y)\,cos(r) & -cos(y)\,sin(p) - sin(y)\,sin(r)\,cos(p) \\ \sin(y)\cos(p) - \cos(y)\sin(r)\sin(p) & -\cos(y)\cos(r) & -sin(y)\,sin(p) + cos(y)\,sin(r)\,cos(p) \\ -\cos(r)\sin(p) & -\sin(r) & -\cos(r)\cos(p) \end{bmatrix} \quad (9.13)$$

Where y is the Yaw, r is the $Roll$, and p is the $Pitch$.

Finally we can re-write the relations between the two frames using the transformation matrix stated in equation (9.13). These relations are given in equations (9.14), (9.15), and (9.16) for the $X, Y, and\ Z$ coordinates.

$$X_t = X_u(sin\,y\,sin\,r\,sin\,p - cos\,y\,cos\,p) + Y_u(cos\,y\,sin\,r\,sin\,p - sin\,y\,cos\,p) + Z_u(cos\,r\,sin\,p) \quad (9.14)$$

$$Y_t = -X_u(sin\,y\,cos\,r) + Y_u(cos\,y\cos r) + Z_u(sin\,r) \quad (9.15)$$

$$Z_t = X_u(cos\,y\,sin\,p + sin\,y\,sin\,r\,cos\,p) + Y_u(sin\,y\,sin\,p - cos\,y\,sin\,r\,cos\,p) + Z_u(cos\,r\,cos\,p) \quad (9.16)$$

This concludes the coordinate transformation from the ground transmitter centered frame to the UAV centered frame.

References

[1] L. Wang, Y. Li, H. Zhu, and L. Shen, "Target state estimation and prediction based standoff tracking of ground moving target using a fixed-wing UAV," in *Control and Automation (ICCA), 2010 8th IEEE International Conference on*, 2010, pp. 273–278.

[2] O. Meister, N. Frietsch, C. Ascher, and G. F. Trommer, "Adaptive path planning for VTOL-UAVs," *IEEE Aerospace and Electronic Systems Magazine*, vol. 24, no. 7, pp. 36 –41, Jul. 2009.

[3] N. Regina and M. Zanzi, "UAV guidance law for ground-based target trajectory tracking and loitering," in *2011 IEEE Aerospace Conference*, 2011, pp. 1 –9.

[4] J. Min, Y. Jeong, and I. S. Kweon, "Robust visual lock-on and simultaneous localization for an unmanned aerial vehicle," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2010, pp. 93 –100.

[5] G. Conte, M. Hempel, P. Rudol, D. Lundström, S. Duranti, M. Wzorek, and P. Doherty, "High accuracy ground target geo-location using autonomous micro aerial vehicle platforms," in *Proceedings of the AIAA-08 Guidance, Navigation, and Control Conference*, 2008.

[6] M. J. Monda, C. A. Woolsey, and C. K. Reddy, "Ground target localization and tracking in a riverine environment from a uav with a gimbaled camera," in *Proceedings of AIAA Guidance, Navigation and Control Conference*, 2007, pp. 6747–6750.

[7] S. S. Ponda, R. M. Kolacinski, and E. Frazzoli, "Trajectory optimization for target localization using small unmanned aerial vehicles," in *AIAA Conf. on Guidance, Navigation, and Control,(Chicago, IL)*, 2009.

[8] A. B. Gershman, Lj. Stanković, and V. Katkovnik, "Sensor array signal tracking using a data-driven window approach," *Signal Processing*, vol. 80, no. 12, pp. 2507–2515, Dec. 2000.

[9] B. R. Huber, "RADIO DETERMINATION ON MINI-UAV PLATFORMS: TRACKING AND LOCATING RADIO TRANSMITTERS," Brigham Young University, 2009.

[10] W. Tidd, R. J. Weber, Y. Huang, and Y. Zhao, “A RF source localization and tracking system,” in *MILITARY COMMUNICATIONS CONFERENCE, 2010-MILCOM 2010*, 2010, pp. 858–863.

[11] I. Amundson, X. Koutsoukos, J. Sallai, and A. Ledeczi, “Mobile Sensor Navigation Using Rapid RF-Based Angle of Arrival Localization,” 2011, pp. 316–325.

[12] E.-E.-L. Lau and W.-Y. Chung, “Enhanced RSSI-Based Real-Time User Location Tracking System for Indoor and Outdoor Environments,” 2007, pp. 1213–1218.

[13] İ. Güvenç, “Enhancements to RSS based indoor tracking systems using kalman filters,” University of New Mexico, 2003.

[14] U. Bandara, M. Hasegawa, M. Inoue, H. Morikawa, and T. Aoyama, “Design and implementation of a bluetooth signal strength based location sensing system,” in *Radio and Wireless Conference, 2004 IEEE*, 2004, pp. 319–322.

[15] G. Tuna, V. C. Gungor, and K. Gulez, “GPS aided Extended Kalman Filter based localization for unmanned vehicles,” in *Signal Processing and Communications Applications Conference (SIU), 2012 20th*, 2012, pp. 1 –4.

[16] M. S. Sharawi, D. N. Aloi, and O. A. Rawashdeh, “Design and Implementation of Embedded Printed Antenna Arrays in Small UAV Wing Structures,” *IEEE Transactions on Antennas and Propagation*, vol. 58, no. 8, pp. 2531–2538, Aug. 2010.

[17] D. M. Moorehouse and A. Humen, *Improved UAV Datalink Performance Using Embedded Antennas*. Tech. Maryland: Nurad Technologies. Web, 2012.

[18] L. Gezer, R. Broadston, D. Jenn, and G. Burgstaller, “Digital tracking array using off-the-shelf hardware,” *Antennas and Propagation Magazine, IEEE*, vol. 50, no. 1, pp. 108–114, 2008.

[19] P. Guttorp and R. A. Lockhart, “Finding the Location of a Signal: A Bayesian Analysis,” *Journal of the American Statistical Association*, vol. 83, no. 402, p. 322, Jun. 1988.

[20] J. C. Chen, K. Yao, and R. E. Hudson, “Source localization and beamforming,” *Signal Processing Magazine, IEEE*, vol. 19, no. 2, pp. 30–39, 2002.

[21] F. Izquierdo, M. Ciurana, F. Barceló, J. Paradells, and E. Zola, “Performance evaluation of a TOA-based trilateration method to locate terminals in WLAN,” in

Wireless Pervasive Computing, 2006 1st International Symposium on, 2006, pp. 1–6.

[22] "XBee® ZB - Digi International." [Online]. Available: http://www.digi.com/products/wireless-wired-embedded-solutions/zigbee-rf-modules/zigbee-mesh-module/xbee-zb-module#specs. [Accessed: 09-Oct-2012].

[23] "Mini Telemaster Kit V2 from Hobby Lobby." [Online]. Available: http://www.hobby-lobby.com/telemaster_mini_v2_1037035_prd1.htm. [Accessed: 09-Oct-2012].

[24] C. F. Lorenzo and T. T. Hartley, "Structurally Integrated Antenna Concepts for HALE UAVs," 2000.

[25] L. M. Hilliard, J. Mead, R. Rincon, and P. H. Hildebrand, "Lightweight linear broadband antennas enabling small UAV wing systems and space flight nanosat concept," in *Geoscience and Remote Sensing Symposium, 2004. IGARSS'04. Proceedings. 2004 IEEE International*, 2004, vol. 5, pp. 3577–3580.

[26] C. Quintero, M. Jenabi, and T. Brukiewa, "MSAG based MAE-UAV active array antenna," in *Radar Conference, 2001. Proceedings of the 2001 IEEE*, 2001, pp. 393–397.

[27] N. Chamberlain, M. Zawadzki, G. Sadowy, E. Oakes, K. Brown, and R. Hodges, "The UAVSAR phased array aperture," in *Aerospace Conference, 2006 IEEE*, 2006, p. 13–pp.

[28] "MATLAB - The Language of Technical Computing." [Online]. Available: http://www.mathworks.com/products/matlab/. [Accessed: 26-Nov-2012].

[29] ANSYS Inc., "ANSYS HFSS." .

[30] C. A. Balanis, *Antenna Theory: Analysis and Design*, 3rd ed. New Jersey, USA: Wiley Interscience, 2005.

[31] A. A. Masoud, "A Harmonic Potential Approach for Simultaneous Planning and Control of a Generic UAV Platform," *Journal of Intelligent & Robotic Systems*, vol. 65, no. 1, pp. 153–173, 2012.

[32] N. X. Vinh, *Flight Mechanics of High-Performance Aircraft*. Cambridge University Press, 1995.

[33] “Cessna 172.” [Online]. Available: http://www.beyond-aviation.com/. [Accessed: 10-May-2012].

[34] “Aerospaceweb.org | Ask Us - Drag Coefficient & Lifting Line Theory.” [Online]. Available: http://www.aerospaceweb.org/question/aerodynamics/q0184.shtml. [Accessed: 25-Nov-2012].

[35] M. Cavcar, “Stall Speed.” Anadolu University, School of Civil Aviation, Turkey.

[36] “MANEUVERABILITY.” Granite Island Group.

[37] “FlightGear Project,” *FlightGear*. [Online]. Available: http://www.flightgear.org. [Accessed: 19-Nov-2012].

[38] “GNU Public License,” *GNU GENERAL PUBLIC LICENSE*. [Online]. Available: http://www.gnu.org/licenses/gpl.txt. [Accessed: 19-Nov-2012].

[39] “FGFS Based Projects,” *FlightGear*. [Online]. Available: http://www.flightgear.org/Projects/. [Accessed: 19-Nov-2012].

[40] V. Kumar, H. Yong, D. Min, and E. Choi, “Auto landing control for small scale unmanned helicopter with flight gear and HILS,” in *Computer Sciences and Convergence Information Technology (ICCIT), 2010 5th International Conference on*, 2010, pp. 676–681.

[41] D. Munoz, F. Bouchereau, C. Vargas, and R. E. Caldera, *Position Location Techniques and Applications*. United Stated of America: Elsevier Inc., 2009.

[42] K. Yu, I. Sharp, and Y. J. Guo, *Ground Based Wireless Positioning*. United Kingdom: Wiley, 2009.

[43] H. Tsuji, P. Cherntanomwong, and J. Takada, “Experiential Evaluation of Outdoor Radio Source Localization Using Spatial Information of Array,” in *Microwave Conference, 2007. APMC 2007. Asia-Pacific*, 2007, pp. 1–4.

[44] R. K. Martin, A. S. King, R. W. Thomas, and J. Pennington, “Practical limits in RSS-based positioning,” in *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, 2011, pp. 2488–2491.

[45] P. J. M. Havinga, “On the calibration and performance of rss-based localization methods,” 2010.

[46] P. N. Pathirana, A. N. Bishop, and A. V. Savkin, “Localization of mobile transmitters by means of linear state estimation using RSS measurements,” in

Control, Automation, Robotics and Vision, 2008. ICARCV 2008. 10th International Conference on, 2008, pp. 210–213.

[47] R. H. Wu, Y. H. Lee, H. W. Tseng, Y. G. Jan, and M. H. Chuang, "Study of characteristics of RSSI signal," in *Industrial Technology, 2008. ICIT 2008. IEEE International Conference on*, 2008, pp. 1–3.

[48] X. Li, "RSS-based location estimation with unknown pathloss model," *Wireless Communications, IEEE Transactions on*, vol. 5, no. 12, pp. 3626–3633, 2006.

[49] M. Laaraiedh, S. Avrillon, and B. Uguen, "Enhancing positioning accuracy through RSS based ranging and weighted least square approximation," in *Proceedings of the International Conference on Positioning and Context-Awareness*, 2009, pp. 31–35.

[50] R. Storn and K. Price, "Minimizing the real functions of the ICEC'96 contest by differential evolution," in *, Proceedings of IEEE International Conference on Evolutionary Computation, 1996*, 1996, pp. 842 –844.

[51] R. Storn, "On the usage of differential evolution for function optimization," in *Fuzzy Information Processing Society, 1996. NAFIPS. 1996 Biennial Conference of the North American*, 1996, pp. 519 –523.

[52] J. Vesterstrom and R. Thomsen, "A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems," in *Congress on Evolutionary Computation, 2004. CEC2004*, 2004, vol. 2, pp. 1980 – 1987 Vol.2.

[53] G. A. Bakare, G. Krost, G. K. Venayagamoorthy, and U. O. Aliyu, "Comparative application of Differential Evolution and particle swarm techniques to reactive power and voltage control," in *Intelligent Systems Applications to Power Systems, 2007. ISAP 2007. International Conference on*, 2007, pp. 1–6.

[54] "IEEE 802.15.4." [Online]. Available: http://www.ieee802.org/15/pub/TG4.html. [Accessed: 22-Nov-2012].

[55] "Aircraft Rotations." [Online]. Available: http://www.grc.nasa.gov/WWW/K-12/airplane/rotations. [Accessed: 24-Nov-2012].

Vita

Mohamed Adel Ibrahim was born in June 16 1985 in Alexandria, Egypt. He received his B.Sc. degree from Alexandria Institute of Engineering and Technology (AIET) in 2007, and then he moved to Saudi Arabia where he received his M.Sc. degree from King Fahd University of Petroleum and Minerals (KFUPM) in 2013.

Contact Information

Permanent Address: 12 Amr Ebn Elaas street, Miamy 21411, Alexandria, Egypt.

Current Address: P.O. Box 8662, KFUPM Campus, Dhahran 31261, Saudi Arabia.

E-mail: madeli166@gmail.com

Cellphone: +966 591653763